|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-195] [Erroneous results when converting between transformed units](https://java.net/jira/browse/UNITSOFMEASUREMENT-195) Created: 29/Sep/16  Updated: 06/Oct/16  Resolved: 06/Oct/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [1.0](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16917) |
| **Fix Version/s:** | [1.0.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17751) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [omahdi](https://java.net/jira/secure/ViewProfile.jspa?name=omahdi) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Quantities are not converted properly when converting between different transformed units, e.g. converting a quantity with unit Units.METRE to MetricPrefix.CENTI(Units.METRE) does not produce the expected result:  This issue has been reported on [stackoverflow.com](http://stackoverflow.com/questions/39717722/) and can be reproduced with the following code:  package de.vsbw.obi.jtink;  import javax.measure.spi.ServiceProvider;  import javax.measure.Quantity;  import javax.measure.quantity.Length;  import javax.measure.spi.QuantityFactory;  import tec.units.ri.unit.Units;  import tec.units.ri.unit.MetricPrefix;  public class JScienceUnits {  public static void main(String[] args) {  ServiceProvider provider = ServiceProvider.current();  QuantityFactory<Length> lengthFactory = provider.getQuantityFactory(Length.class);    Quantity<Length> q = lengthFactory.create(3, Units.METRE);  System.out.println("q = " + q + ", q.to(CENTI(METRE)) = " + q.to(MetricPrefix.CENTI(Units.METRE)));  }  The output is  q = 5 m, q.to(CENTI(METRE)) = 0.05 cm  instead of the expected  q = 5 m, q.to(CENTI(METRE)) = 500.0 cm.  Further investigation revealed a bug in the doubleValue(Unit<Q>) method of tec.units.ri.quantity.NumberQuantity at [line 124](https://github.com/unitsofmeasurement/unit-ri/blob/1.0/src/main/java/tec/units/ri/quantity/NumberQuantity.java#L124), which erroneously builds a converter for the \***inverse**\* unit transformation. Changing that line to  UnitConverter converter = myUnit.getConverterTo(unit);  seems to fix the issue. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-194] [Prepare Final](https://java.net/jira/browse/UNITSOFMEASUREMENT-194) Created: 07/Aug/16  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200), [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16916) |
| **Fix Version/s:** | [1.0](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16917) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | final, release |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Prepare Final release in JavaDoc and other areas where relevant. |

|  |  |
| --- | --- |
| **[[Clarification] best way to convert Product of units to product of baseUnits](https://java.net/jira/browse/UNITSOFMEASUREMENT-182)** ([UNITSOFMEASUREMENT-182](https://java.net/jira/browse/UNITSOFMEASUREMENT-182)) [UNITSOFMEASUREMENT-193] [What about Dimension.getProductDimensions() ?](https://java.net/jira/browse/UNITSOFMEASUREMENT-193) Created: 21/Jun/16  Updated: 28/Jun/16  Resolved: 28/Jun/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-183~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-183) | getProductUnits should probably be re... | Resolved | |  |
| **Tags:** | design, naming |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| If Unit.getProductUnits() is renamed to getBaseUnits() or similar, would that imply, Dimension.getProductDimensions() should also be renamed or does **Product** make sense in its case? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Jun/16 ] |
| The JavaDoc is oriented on descriptions by e.g. Wikipedia: <https://en.wikipedia.org/wiki/Dimensional_analysis#Mathematical_properties>  Referring to "Fundamental Dimensions" or also "Fundamental Units" (another synonym for "Base Units")  In UCAR they called it "factors" but its API consists of several additional elements, so we cannot directly compare them any more.  <https://en.wikipedia.org/wiki/Physical_quantity> usually speaks of "Base Quantity" (though the dimension is also there) so should we rename getProductDimensions() to either getBaseDimensions() or e.g. getFundamentalDimensions() ? |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-192] [Short number support for Quantities](https://java.net/jira/browse/UNITSOFMEASUREMENT-192) Created: 18/Apr/16  Updated: 21/Jun/16  Resolved: 21/Jun/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | embedded, number |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| See <https://github.com/unitsofmeasurement/uom-systems/issues/29> embedded systems tend to use very small numeric types like Short which has no "wrapper" type in the RI (or SE port) right now. Therefore it is cast to some of the closest equivalent types which leads to differrent results/rounding errors. |

|  |  |
| --- | --- |
| **[Refactoring SPI](https://java.net/jira/browse/UNITSOFMEASUREMENT-152)** ([UNITSOFMEASUREMENT-152](https://java.net/jira/browse/UNITSOFMEASUREMENT-152)) [UNITSOFMEASUREMENT-191] [Could QuantityFactoryService be merged into ServiceProvider, too?](https://java.net/jira/browse/UNITSOFMEASUREMENT-191) Created: 03/Apr/16  Updated: 08/Jun/16  Resolved: 08/Jun/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | factory, quantity, question, refactoring, vote |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At the moment unlike the other 2 service interfaces, QuantityFactoryService only defines getQuantityFactory() for a single, default instance of QuantityFactory (per quantity).  Although this involves some "caching" (depending on implementations) otherwise the service does not offer multiple factories or additional methods, so could we consider QuantityFactory itself a "service" (would not call it \*Service here) and have ServiceProvider directly define getQuantityFactory() instead of getQuantityFactoryService() ? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 11/May/16 ] |
| I would support this simplification proposal. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/May/16 ] |
| Thanks, will speak to @otaviojava this week, too. If neither he nor others object we'll go for it. |

|  |  |
| --- | --- |
| **[Refactoring SPI](https://java.net/jira/browse/UNITSOFMEASUREMENT-152)** ([UNITSOFMEASUREMENT-152](https://java.net/jira/browse/UNITSOFMEASUREMENT-152)) [UNITSOFMEASUREMENT-190] [Create PoC for a refactored ServiceProvider](https://java.net/jira/browse/UNITSOFMEASUREMENT-190) Created: 30/Mar/16  Updated: 01/Apr/16  Resolved: 01/Apr/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | refactoring |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This covers work done in <https://github.com/desruisseaux/unit-api> at the moment. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 31/Mar/16 ] |
| Pull request created: <https://github.com/unitsofmeasurement/unit-api/pull/30> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 31/Mar/16 ] |
| Thanks. I suggest a tiny tweak after merging it. To follow other JSRs like JSON-P in their SPI or some new libraries coming with Java 8 (like Streams) is it OK to call the 2 main static accessor methods of ServiceProvider default() and available()? For setDefault() I think this is OK, other options like init() or configure() (as used by either JSR 354 of CDI in similar SPI elements) probably mean a bit less, but keeping the 2 static methods shorter with only the actual "bean" methods of the instance itself called get...Service() feels a bit cleaner and more "fluent".  I would do the same with the DimensionalModel class in the RI level SPI which has a current() method, too, so these SPI classes provide a similar user experience. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 31/Mar/16 ] |
| I had another thought on the static method names and it turns out, default() or getDefault() isn't really the right term if you look at it from a modular angle. As the API provides no fallback like Bootstrap did earlier, the "default" provider would be that offered by an implementation like the RI. Then if modules add their own providers with the right priority, they override that "default" and the "active" or "current" one is no longer the default. Let's try CDI's current() I think that puts it best. |

|  |  |
| --- | --- |
| **[Refactoring SPI](https://java.net/jira/browse/UNITSOFMEASUREMENT-152)** ([UNITSOFMEASUREMENT-152](https://java.net/jira/browse/UNITSOFMEASUREMENT-152)) [UNITSOFMEASUREMENT-189] [Should "Bootstrap" be final or abstract?](https://java.net/jira/browse/UNITSOFMEASUREMENT-189) Created: 22/Mar/16  Updated: 01/Apr/16  Resolved: 01/Apr/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-152~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-152) | Refactoring SPI | Resolved | |  |
| **Tags:** | design, question |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Regardless of the name and functionality it offers, it's a valid question, if (the class currently known as) Bootstrap should be final as it is right now, making it the de-facto only <b>Singleton</b> for the API or if there was a need to sub-class it for some use cases?  CDI is a good example although it is far more complex than what we need and our users may expect from the API and SPI, but one could imagine  Bootstrap b = Bootstrap.current();  List<SystemOfUnitsService> systemOfUnitsServices = b.getServices(SystemOfUnitsService.class);  ...  Whether the name was Bootstrap, ServiceProvider (along the lines of JSONP: <https://json-processing-spec.java.net/nonav/releases/1.0/pfd-draft/javadocs/javax/json/spi/JsonProvider.html>) is subject to <https://java.net/jira/browse/UNITSOFMEASUREMENT-152> and not directly relevant to this task.  It could add more flexibility, but create an overhead having to call Bootstrap.current() (or default()) or similar first rather than just accessing static methods of Bootstrap. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 30/Mar/16 ] |
| I think this needs to be dealt with and documented, also see some later discussions in <https://java.net/jira/browse/UNITSOFMEASUREMENT-152>. If we require each module to define at least one ServiceProvider, then in most cases the implementation level ServiceProvider should be taken as super-class, not ServiceProvider itself. This way modules are free to decide what to extend/customize and what to leave up to the implementation, e.g. QuantityFactory or UnitFormat instances if they have no need to declare their own. Unlike the current tec.units.ri.internal.RIServiceProvider it should therefore simply move to tec.units.ri.spi allowing extension by modules. It was "internal" since only Bootstrap/ServiceLoader had to know about it, otherwise it would be opaque and an internal implementation detail. If we offer it to other modules, that has to change.  Each module may declare its "leaf" ServiceLoader class final if they want to prohibit extensions, otherwise keep it open to additional modules to extend. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Apr/16 ] |
| As it was merged with ServiceProvider that needs to be abstract by nature to allow extension. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-188] [Comparator in RIServiceProvider](https://java.net/jira/browse/UNITSOFMEASUREMENT-188) Created: 22/Mar/16  Updated: 01/May/16  Resolved: 01/May/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| In the SE Port uom-se, its PriorityAwareServiceProvider uses  Priority prio1Annot = o1.getClass().getAnnotation(Priority.class);  via the @Priority annotation to sort services of a particular kind. That way e.g. extension modules can decide, if they want their own version of a service or that by the underlying implementation to be default or have a higher priority.  Since getAnnotation() is not available on Java ME, we dropped this entirely from the RI. However, sorting by "some" criteria would be beneficial, but we have to make it available and use it under ME or leave this as a known limitation of the Reference Implementation. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/May/16 ] |
| This became irrelevant with refactoring of ServiceProvider which allows to use getPriority() and does not require extra annotations. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-187] [uom-se doesn't compile with JDK 8 (update 73,74)](https://java.net/jira/browse/UNITSOFMEASUREMENT-187) Created: 21/Mar/16  Updated: 21/Mar/16  Resolved: 21/Mar/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915), [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16916) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Critical |
| **Reporter:** | [lfoppiano](https://java.net/jira/secure/ViewProfile.jspa?name=lfoppiano) | **Assignee:** | Unassigned |
| **Resolution:** | Invalid | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| I'm trying to use the uom-se with JDK 8 but I'm facing some problem while compiling. I've seen similar problems were already raised and solved before but I could not find the cause of why it is still not compiling.  I'm really not sure whether it is a problem on my local machine or something due to the JDK 1.8.  I'm attaching the maven log:  Johan:uom-se lfoppiano$ mvn clean install  [INFO] Scanning for projects...  [INFO]  [INFO] ------------------------------------------------------------------------  [INFO] Building Units of Measurement Implementation for Java SE 0.8  [INFO] ------------------------------------------------------------------------  [INFO]  [INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ uom-se ---  [INFO] Deleting /Users/lfoppiano/development/inria/unit-normalization/uom-se/target  [INFO]  [INFO] --- maven-enforcer-plugin:1.2:enforce (enforce-maven) @ uom-se ---  [INFO]  [INFO] --- maven-resources-plugin:2.7:resources (default-resources) @ uom-se ---  [INFO] Using 'UTF-8' encoding to copy filtered resources.  [INFO] Copying 16 resources  [INFO]  [INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ uom-se ---  [INFO] Changes detected - recompiling the module!  [INFO] Compiling 83 source files to /Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/AbstractUnitFormat.java]]  [parsing completed 16ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/spi/Range.java]]  [parsing completed 3ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/ProductElement.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/Token.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/time/TimeQuantities.java]]  [parsing completed 3ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/QuantityFunctions.java]]  [parsing completed 8ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/package-info.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java]]  [parsing completed 8ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/time/TimedData.java]]  [parsing completed 3ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/QuantitySupplier.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/DefaultQuantityFormat.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/DefaultSystemOfUnitsService.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/BinaryPrefix.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/AlternateUnit.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/MinimumSupplier.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/Quantities.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/MetricPrefix.java]]  [parsing completed 3ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/package-info.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/DefaultUnitFormatService.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/DecimalQuantity.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/spi/StandardModel.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/l10n/ResourceBundleEnumeration.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/BaseUnit.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/ExpConverter.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/QuantitySummaryStatistics.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/DefaultQuantityFactory.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/LocalUnitFormatParser.java]]  [parsing completed 8ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/NaturalOrder.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/spi/DimensionalModel.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/ComparableQuantity.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/DefaultCharStream.java]]  [parsing completed 4ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/DescriptionSupplier.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/Parser.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/l10n/BundleToMapAdapter.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java]]  [parsing completed 9ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/MultiplyConverter.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/PiDivisorConverter.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/PiMultiplierConverter.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/quantity/DefaultQuantityFactoryService.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/AbstractConverter.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/quantity/OldProxyQuantityFactory.java]]  [parsing completed 4ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/UnitFormatParser.java]]  [parsing completed 6ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/LogConverter.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/AbstractQuantity.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/RationalConverter.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/TokenException.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/EBNFUnitFormat.java]]  [parsing completed 3ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/MaximumSupplier.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/ValueSupplier.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/PriorityAwareServiceProvider.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/QuantityFormat.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/AbstractSystemOfUnits.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/OutputHelper.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/Nameable.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/spi/Measurement.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/UnitTokenManager.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/QuantityDimension.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/NumberQuantity.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/TransformedUnit.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/NumberSpaceQuantityFormat.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/ProductUnit.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/package-info.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/DoubleQuantity.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/Converter.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/UnitTokenConstants.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/UnitSupplier.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/l10n/NumberFormat2.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/DoubleFactorSupplier.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/FormatBehavior.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/time/TimeUnitQuantity.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/AnnotatedUnit.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/QuantityRange.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/QuantityConverter.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/UnitConverterSupplier.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/l10n/MultiPropertyResourceBundle.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/LocalUnitFormat.java]]  [parsing completed 3ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SymbolMap.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/function/AddConverter.java]]  [parsing completed 0ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/internal/format/TokenMgrError.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/time/TemporalQuantity.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/spi/AbstractMeasurement.java]]  [parsing completed 1ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/AbstractUnit.java]]  [parsing completed 2ms]  [parsing started RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/quantity/ProxyQuantityFactory.java]]  [parsing completed 1ms]  [search path for source files: /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java,/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/generated-sources/annotations]  [search path for class files: /Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/resources.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/rt.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/sunrsasign.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/jsse.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/jce.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/charsets.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/jfr.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/classes,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/cldrdata.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/dnsns.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/jfxrt.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/localedata.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/nashorn.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/sunec.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/sunjce\_provider.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/sunpkcs11.jar,/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/jre/lib/ext/zipfs.jar,/System/Library/Java/Extensions/MRJToolkit.jar,/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes,/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar,/Users/lfoppiano/.m2/repository/javax/annotation/javax.annotation-api/1.2/javax.annotation-api-1.2.jar,.]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/IOException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/text/ParsePosition.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/Unit.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/format/ParserException.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/format/UnitFormat.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Object.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/UnitConverter.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/Field.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Enumeration.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/HashMap.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Map.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/ResourceBundle.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/TreeMap.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/logging/Level.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/logging/Logger.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/SuppressWarnings.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Appendable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/String.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/CharSequence.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/IllegalArgumentException.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Dimensionless.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/Serializable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/math/BigInteger.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/ParameterizedType.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/Type.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Comparable.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/Quantity.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/StringBuilder.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Enum.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/FunctionalInterface.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/Dimension.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Objects.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Class.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Integer.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/UnconvertibleException.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/IncommensurableException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/math/BigDecimal.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/math/MathContext.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/DoubleSupplier.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/Supplier.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/ArrayList.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/List.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Number.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Double.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/ArithmeticException.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Acceleration.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/AmountOfSubstance.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Angle.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Area.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/CatalyticActivity.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/ElectricCapacitance.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/ElectricCharge.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/ElectricConductance.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/ElectricCurrent.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/ElectricInductance.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/ElectricPotential.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/ElectricResistance.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Energy.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Force.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Frequency.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Illuminance.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Length.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/LuminousFlux.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/LuminousIntensity.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/MagneticFlux.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/MagneticFluxDensity.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Mass.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Power.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Pressure.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/RadiationDoseAbsorbed.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/RadiationDoseEffective.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Radioactivity.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/SolidAngle.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Speed.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Temperature.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Time.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/quantity/Volume.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/spi/SystemOfUnits.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/stream/Collectors.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/LocalTime.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/temporal/ChronoUnit.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/temporal/Temporal.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/temporal/TemporalAdjuster.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/temporal/TemporalUnit.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/concurrent/TimeUnit.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/Duration.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/temporal/TemporalAmount.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Comparator.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Set.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/ClassCastException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/UnsupportedOperationException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/BinaryOperator.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/Function.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/Predicate.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/stream/Collector.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/Instant.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/text/FieldPosition.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/text/Format.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/text/NumberFormat.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Cloneable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/StringBuffer.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Collection.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/concurrent/ConcurrentHashMap.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/spi/SystemOfUnitsService.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/annotation/javax.annotation-api/1.2/javax.annotation-api-1.2.jar(javax/annotation/Priority.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/spi/UnitFormatService.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Locale.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Iterator.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/NoSuchElementException.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/spi/QuantityFactory.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/Reader.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/MeasurementException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/RuntimeException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Exception.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Throwable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/InputStream.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/PrintStream.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/UnsupportedEncodingException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/math/RoundingMode.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/spi/QuantityFactoryService.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/InvocationHandler.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/Method.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/Proxy.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Error.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/StrictMath.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/StringReader.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Formattable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Formatter.class)]]  [loading ZipFileIndexFileObject[/Users/lfoppiano/.m2/repository/javax/measure/unit-api/0.8/unit-api-0.8.jar(javax/measure/spi/ServiceProvider.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Collections.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/ServiceLoader.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/PropertyResourceBundle.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Vector.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/annotation/Target.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/annotation/ElementType.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/annotation/Retention.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/annotation/RetentionPolicy.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/annotation/Annotation.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Override.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/SafeVarargs.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Deprecated.class)]]  [checking tec.uom.se.format.AbstractUnitFormat]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/AutoCloseable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/AbstractStringBuilder.class)]]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/format/AbstractUnitFormat.class]]  [checking tec.uom.se.format.SymbolMap]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/GenericDeclaration.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/AnnotatedElement.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Byte.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Character.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Short.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Long.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Float.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Boolean.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Void.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/SortedMap.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/AbstractMap.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/NoSuchFieldException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/SecurityException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/AccessibleObject.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/Member.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Iterable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/ClassNotFoundException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/ReflectiveOperationException.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/IllegalAccessException.class)]]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/format/SymbolMap.class]]  [checking tec.uom.se.AbstractUnit]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Math.class)]]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/AbstractUnit.class]]  [checking tec.uom.se.unit.MetricPrefix]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/CloneNotSupportedException.class)]]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/unit/MetricPrefix.class]]  [checking tec.uom.se.function.UnitConverterSupplier]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/function/UnitConverterSupplier.class]]  [checking tec.uom.se.unit.AnnotatedUnit]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/unit/AnnotatedUnit.class]]  [checking tec.uom.se.function.RationalConverter]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/AbstractList.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/AbstractCollection.class)]]  [checking tec.uom.se.AbstractConverter]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/function/RationalConverter.class]]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/AbstractConverter$Identity.class]]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/AbstractConverter$Pair.class]]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/AbstractConverter$1.class]]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/AbstractConverter.class]]  [checking tec.uom.se.function.Converter]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/function/Converter.class]]  [checking tec.uom.se.function.ValueSupplier]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/function/ValueSupplier.class]]  [checking tec.uom.se.spi.Range]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/spi/Range.class]]  [checking tec.uom.se.function.MinimumSupplier]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/function/MinimumSupplier.class]]  [checking tec.uom.se.function.MaximumSupplier]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/function/MaximumSupplier.class]]  [checking tec.uom.se.unit.ElementProduct]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/unit/ElementProduct.class]]  [checking tec.uom.se.internal.format.Token]  [wrote RegularFileObject[/Users/lfoppiano/development/inria/unit-normalization/uom-se/target/classes/tec/uom/se/internal/format/Token.class]]  [checking tec.uom.se.unit.Units]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/HashSet.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/AbstractSet.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/stream/Stream.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/stream/BaseStream.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/BiConsumer.class)]]  [checking tec.uom.se.AbstractSystemOfUnits]  [checking tec.uom.se.function.Nameable]  [checking tec.uom.se.quantity.time.TimeQuantities]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/time/temporal/TemporalAccessor.class)]]  [checking tec.uom.se.unit.TransformedUnit]  [checking tec.uom.se.quantity.time.TimeUnitQuantity]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/IllegalStateException.class)]]  [checking tec.uom.se.quantity.time.TemporalQuantity]  [checking tec.uom.se.AbstractQuantity]  [checking tec.uom.se.ComparableQuantity]  [checking tec.uom.se.function.UnitSupplier]  [checking tec.uom.se.unit.BaseUnit]  [checking tec.uom.se.unit.AlternateUnit]  [checking tec.uom.se.unit.ProductUnit]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/System.class)]]  [checking tec.uom.se.function.QuantityFunctions]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/BiFunction.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/stream/Collector$Characteristics.class)]]  [checking tec.uom.se.function.QuantitySummaryStatistics]  [checking tec.uom.se.quantity.time.TimedData]  [checking tec.uom.se.function.QuantitySupplier]  [checking tec.uom.se.format.DefaultQuantityFormat]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/NullPointerException.class)]]  [checking tec.uom.se.format.QuantityFormat]  [checking tec.uom.se.format.Parser]  [checking tec.uom.se.format.NumberSpaceQuantityFormat]  [checking tec.uom.se.format.FormatBehavior]  [checking tec.uom.se.internal.DefaultSystemOfUnitsService]  [checking tec.uom.se.unit.BinaryPrefix]  [checking tec.uom.se.quantity.Quantities]  [checking tec.uom.se.format.SimpleUnitFormat]  [checking tec.uom.se.internal.format.DefaultUnitFormatService]  [checking tec.uom.se.function.MultiplyConverter]  [checking tec.uom.se.function.DoubleFactorSupplier]  [checking tec.uom.se.quantity.DecimalQuantity]  [checking tec.uom.se.spi.StandardModel]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/Map$Entry.class)]]  [checking tec.uom.se.spi.DimensionalModel]  [checking tec.uom.se.internal.format.l10n.ResourceBundleEnumeration]  [checking tec.uom.se.function.ExpConverter]  [checking tec.uom.se.quantity.DefaultQuantityFactory]  [checking tec.uom.se.internal.format.UnitTokenConstants]  [checking tec.uom.se.internal.format.LocalUnitFormatParser]  [checking tec.uom.se.internal.format.TokenException]  [checking tec.uom.se.internal.format.UnitTokenManager]  [checking tec.uom.se.internal.format.DefaultCharStream]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Readable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/Closeable.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/InputStreamReader.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/nio/charset/CharsetDecoder.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/nio/charset/Charset.class)]]  [checking tec.uom.se.function.NaturalOrder]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/ToDoubleFunction.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/ToLongFunction.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/function/ToIntFunction.class)]]  [checking tec.uom.se.function.DescriptionSupplier]  [checking tec.uom.se.internal.format.l10n.BundleToMapAdapter]  [checking tec.uom.se.function.PiDivisorConverter]  [checking tec.uom.se.function.PiMultiplierConverter]  [checking tec.uom.se.internal.quantity.DefaultQuantityFactoryService]  [checking tec.uom.se.internal.quantity.OldProxyQuantityFactory]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/concurrent/ConcurrentMap.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/ClassLoader.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/reflect/Executable.class)]]  [checking tec.uom.se.internal.format.UnitFormatParser]  [checking tec.uom.se.function.LogConverter]  [checking tec.uom.se.format.EBNFUnitFormat]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/ResourceBundle$Control.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/Flushable.class)]]  [checking tec.uom.se.internal.PriorityAwareServiceProvider]  [checking tec.uom.se.internal.format.OutputHelper]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/FilterOutputStream.class)]]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/io/OutputStream.class)]]  [checking tec.uom.se.spi.AbstractMeasurement]  [checking tec.uom.se.spi.Measurement]  [checking tec.uom.se.quantity.QuantityDimension]  [checking tec.uom.se.quantity.NumberQuantity]  [checking tec.uom.se.quantity.DoubleQuantity]  [checking tec.uom.se.internal.format.l10n.NumberFormat2]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/NumberFormatException.class)]]  [checking tec.uom.se.quantity.QuantityRange]  [checking tec.uom.se.function.QuantityConverter]  [checking tec.uom.se.internal.format.l10n.MultiPropertyResourceBundle]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/util/RandomAccess.class)]]  [checking tec.uom.se.format.LocalUnitFormat]  [loading ZipFileIndexFileObject[/Library/Java/JavaVirtualMachines/jdk1.8.0\_31.jdk/Contents/Home/lib/ct.sym(META-INF/sym/rt.jar/java/lang/Package.class)]]  [checking tec.uom.se.function.AddConverter]  [checking tec.uom.se.internal.format.TokenMgrError]  [checking tec.uom.se.quantity.ProxyQuantityFactory]  [total 1249ms]  [INFO] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/LocalUnitFormat.java: Some input files use or override a deprecated API.  [INFO] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/LocalUnitFormat.java: Recompile with -Xlint:deprecation for details.  [INFO] -------------------------------------------------------------  [ERROR] COMPILATION ERROR :  [INFO] -------------------------------------------------------------  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[102,34] incompatible types: inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Dimensionless,Q  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[166,22] incompatible types: inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Length,Q  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[176,22] incompatible types: inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.AmountOfSubstance,Q  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[187,22] incompatible types: inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Time,Q  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[425,22] incompatible types: inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Speed,Q  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[432,22] incompatible types: inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Acceleration,Q  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[439,22] incompatible types: inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Area,Q  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[501,11] no suitable method found for addUnit(tec.uom.se.unit.TransformedUnit<javax.measure.quantity.Angle>)  method tec.uom.se.AbstractSystemOfUnits.<U>addUnit(U,java.lang.String) is not applicable  (cannot infer type-variable(s) U  (actual and formal argument lists differ in length))  method tec.uom.se.unit.Units.<U>addUnit(U) is not applicable  (inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Angle  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  method tec.uom.se.unit.Units.<U>addUnit(U,java.lang.Class<? extends javax.measure.Quantity<?>>) is not applicable  (cannot infer type-variable(s) U  (actual and formal argument lists differ in length))  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[508,11] no suitable method found for addUnit(tec.uom.se.unit.TransformedUnit<javax.measure.quantity.Angle>)  method tec.uom.se.AbstractSystemOfUnits.<U>addUnit(U,java.lang.String) is not applicable  (cannot infer type-variable(s) U  (actual and formal argument lists differ in length))  method tec.uom.se.unit.Units.<U>addUnit(U) is not applicable  (inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Angle  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  method tec.uom.se.unit.Units.<U>addUnit(U,java.lang.Class<? extends javax.measure.Quantity<?>>) is not applicable  (cannot infer type-variable(s) U  (actual and formal argument lists differ in length))  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[515,11] no suitable method found for addUnit(tec.uom.se.unit.TransformedUnit<javax.measure.quantity.Angle>)  method tec.uom.se.AbstractSystemOfUnits.<U>addUnit(U,java.lang.String) is not applicable  (cannot infer type-variable(s) U  (actual and formal argument lists differ in length))  method tec.uom.se.unit.Units.<U>addUnit(U) is not applicable  (inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Angle  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  method tec.uom.se.unit.Units.<U>addUnit(U,java.lang.Class<? extends javax.measure.Quantity<?>>) is not applicable  (cannot infer type-variable(s) U  (actual and formal argument lists differ in length))  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[522,11] no suitable method found for addUnit(tec.uom.se.unit.TransformedUnit<javax.measure.quantity.Volume>)  method tec.uom.se.AbstractSystemOfUnits.<U>addUnit(U,java.lang.String) is not applicable  (cannot infer type-variable(s) U  (actual and formal argument lists differ in length))  method tec.uom.se.unit.Units.<U>addUnit(U) is not applicable  (inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  method tec.uom.se.unit.Units.<U>addUnit(U,java.lang.Class<? extends javax.measure.Quantity<?>>) is not applicable  (cannot infer type-variable(s) U  (actual and formal argument lists differ in length))  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[561,47] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Length>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Length  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Length  upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1127,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1128,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1129,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1130,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1131,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1132,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1133,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1134,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [INFO] 20 errors  [INFO] -------------------------------------------------------------  [INFO] ------------------------------------------------------------------------  [INFO] BUILD FAILURE  [INFO] ------------------------------------------------------------------------  [INFO] Total time: 2.441 s  [INFO] Finished at: 2016-03-21T09:16:24+01:00  [INFO] Final Memory: 19M/308M  [INFO] ------------------------------------------------------------------------  [ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.3:compile (default-compile) on project uom-se: Compilation failure: Compilation failure:  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[102,34] incompatible types: inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Dimensionless,Q  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[166,22] incompatible types: inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Length,Q  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[176,22] incompatible types: inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.AmountOfSubstance,Q  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[187,22] incompatible types: inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Time,Q  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[425,22] incompatible types: inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Speed,Q  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[432,22] incompatible types: inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Acceleration,Q  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[439,22] incompatible types: inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Area,Q  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[501,11] no suitable method found for addUnit(tec.uom.se.unit.TransformedUnit<javax.measure.quantity.Angle>)  [ERROR] method tec.uom.se.AbstractSystemOfUnits.<U>addUnit(U,java.lang.String) is not applicable  [ERROR] (cannot infer type-variable(s) U  [ERROR] (actual and formal argument lists differ in length))  [ERROR] method tec.uom.se.unit.Units.<U>addUnit(U) is not applicable  [ERROR] (inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Angle  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.unit.Units.<U>addUnit(U,java.lang.Class<? extends javax.measure.Quantity<?>>) is not applicable  [ERROR] (cannot infer type-variable(s) U  [ERROR] (actual and formal argument lists differ in length))  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[508,11] no suitable method found for addUnit(tec.uom.se.unit.TransformedUnit<javax.measure.quantity.Angle>)  [ERROR] method tec.uom.se.AbstractSystemOfUnits.<U>addUnit(U,java.lang.String) is not applicable  [ERROR] (cannot infer type-variable(s) U  [ERROR] (actual and formal argument lists differ in length))  [ERROR] method tec.uom.se.unit.Units.<U>addUnit(U) is not applicable  [ERROR] (inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Angle  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.unit.Units.<U>addUnit(U,java.lang.Class<? extends javax.measure.Quantity<?>>) is not applicable  [ERROR] (cannot infer type-variable(s) U  [ERROR] (actual and formal argument lists differ in length))  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[515,11] no suitable method found for addUnit(tec.uom.se.unit.TransformedUnit<javax.measure.quantity.Angle>)  [ERROR] method tec.uom.se.AbstractSystemOfUnits.<U>addUnit(U,java.lang.String) is not applicable  [ERROR] (cannot infer type-variable(s) U  [ERROR] (actual and formal argument lists differ in length))  [ERROR] method tec.uom.se.unit.Units.<U>addUnit(U) is not applicable  [ERROR] (inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Angle  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.unit.Units.<U>addUnit(U,java.lang.Class<? extends javax.measure.Quantity<?>>) is not applicable  [ERROR] (cannot infer type-variable(s) U  [ERROR] (actual and formal argument lists differ in length))  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[522,11] no suitable method found for addUnit(tec.uom.se.unit.TransformedUnit<javax.measure.quantity.Volume>)  [ERROR] method tec.uom.se.AbstractSystemOfUnits.<U>addUnit(U,java.lang.String) is not applicable  [ERROR] (cannot infer type-variable(s) U  [ERROR] (actual and formal argument lists differ in length))  [ERROR] method tec.uom.se.unit.Units.<U>addUnit(U) is not applicable  [ERROR] (inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.unit.Units.<U>addUnit(U,java.lang.Class<? extends javax.measure.Quantity<?>>) is not applicable  [ERROR] (cannot infer type-variable(s) U  [ERROR] (actual and formal argument lists differ in length))  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/unit/Units.java:[561,47] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Length>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Length  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Length  [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1127,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1128,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1129,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1130,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1131,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1132,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1133,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] /Users/lfoppiano/development/inria/unit-normalization/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1134,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.AbstractUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q>)  [ERROR] -> [Help 1]  [ERROR]  [ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.  [ERROR] Re-run Maven using the -X switch to enable full debug logging.  [ERROR]  [ERROR] For more information about the errors and possible solutions, please read the following articles:  [ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [lfoppiano](https://java.net/jira/secure/ViewProfile.jspa?name=lfoppiano) [ 21/Mar/16 ] |
| I'm sorry, I've just realised I should have opened this task on github. I've done that here (<https://github.com/unitsofmeasurement/uom-se/issues/115>). You can close this ticket. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 21/Mar/16 ] |
| Filed by mistake using an older JDK |

|  |  |
| --- | --- |
| **[Refactoring SPI](https://java.net/jira/browse/UNITSOFMEASUREMENT-152)** ([UNITSOFMEASUREMENT-152](https://java.net/jira/browse/UNITSOFMEASUREMENT-152)) [UNITSOFMEASUREMENT-186] [Change ServiceProvider.available() to Collection](https://java.net/jira/browse/UNITSOFMEASUREMENT-186) Created: 19/Mar/16  Updated: 15/Jun/16  Resolved: 15/Jun/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-152~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-152) | Refactoring SPI | Resolved | |  |
| **Tags:** | Collection, design |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| ServiceProvider.getAvailables() returns an array of ServiceProvider classes. It might be more convenient while not affecting e.g. current for loops to use a Collection or subtypes such as List (maybe better than Set although we probably won't expect duplicate providers there either). |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-185] [Non-functional Requirements (mostly for RI)](https://java.net/jira/browse/UNITSOFMEASUREMENT-185) Created: 17/Mar/16  Updated: 04/Oct/16  Resolved: 04/Oct/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.0.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17751) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Trivial |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-97](https://java.net/jira/browse/UNITSOFMEASUREMENT-97) | [Add a chapter about performance](https://java.net/jira/browse/UNITSOFMEASUREMENT-97) | Sub-task | Resolved |  | | [UNITSOFMEASUREMENT-184](https://java.net/jira/browse/UNITSOFMEASUREMENT-184) | [Loss of precision by using double ins...](https://java.net/jira/browse/UNITSOFMEASUREMENT-184) | Sub-task | Resolved | keilw | |  |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Final Release, Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Mostly for the RI-Guide (unless we face any of it that was good to mention in the Spec, too)  See a possible list of Non-functional Requirements :<http://users.csc.calpoly.edu/~jdalbey/SWE/QA/nonfunctional.html> |

|  |  |
| --- | --- |
| **[Non-functional Requirements (mostly for RI)](https://java.net/jira/browse/UNITSOFMEASUREMENT-185)** ([UNITSOFMEASUREMENT-185](https://java.net/jira/browse/UNITSOFMEASUREMENT-185)) [UNITSOFMEASUREMENT-184] [Loss of precision by using double instead of BigDecimal](https://java.net/jira/browse/UNITSOFMEASUREMENT-184) Created: 17/Mar/16  Updated: 04/Oct/16  Resolved: 04/Oct/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [1.0.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17751) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [lfoppiano](https://java.net/jira/secure/ViewProfile.jspa?name=lfoppiano) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-97~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-97) | Add a chapter about performance | Resolved | |  |
| **Sprint:** | Final Release, Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| I've noticed the following problem, when converting complex units to their bases it can happen to obtains incorrect results due to loss of precision.  The unit-ri classes are based on double primitives, which are still sensible of slight imprecision because of the way they are stored (see link below)  Example: convert a simple value like 0.39 g, to its base unit (kg) result in a 0.0039000000000000005.  A sample code like this:  UnitFormatService formatService = Bootstrap.getService(UnitFormatService.class);  UnitFormat defaultFormatService = formatService.getUnitFormat();  TransformedUnit unit = (TransformedUnit) defaultFormatService.parse("g");  System.out.println(unit.getSystemConverter().convert(0.39));  System.out.println(((AbstractConverter)unit.getSystemConverter()).convert(new BigDecimal("0.39"))); //BigDecimal implementation  unit = (TransformedUnit) defaultFormatService.parse("%");  System.out.println(unit.getSystemConverter().convert(0.009));  System.out.println(((AbstractConverter)unit.getSystemConverter()).convert(new BigDecimal("0.009"))); //BigDecimal implementation  Would return:  3.9000000000000005E-4  0.00039  8.999999999999999E-5  0.00009  It was a new problem for me, in fact, and I've done some research on the subject <http://stackoverflow.com/questions/322749/retain-precision-with-double-in-java> and  I think this kind of library should aim to the maximum precision possible.  My proposal:  1. add a convert(BigDecimal xy) in the UnitConverter interface 2. remove the method convert(Number xy) as it's hiding a conversion using double (which is already present).  I'm asking here before sending code (I have done already the changes for testing purposes), as I'm open to discussion and, maybe, alternative solutions. Once we agreed I will create a pull request on github  Thank you in advance |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Mar/16 ] |
| Thanks for your suggestions. However, convert(BigDecimal) is not acceptable in the API of JSR 363 which aims at the greatest possible number of platforms including and especially Java ME 8 Embedded. BigDecimal or BigInteger do not exist on Java ME, so both API and RI (which runs on both Java ME 8 Embedded and Java SE 7 or higher) obey to those platform constraints.  Like JSR 107 RI is not a fully fledged implementation of the JCache API like e.g. EZCache, Hazelcast XYZ or other (commercial) implementations are, or Glassfish does not provide all aspects of scalability, failover or production guarantees you'll find in WebLogic, WebSphere, JBoss EAP or other commercial Java EE servers, the RI does not intend to be the only implementation for JSR 363. In fact, we created a full implementation for Java SE 8: <https://github.com/unitsofmeasurement/uom-se> please check out your examples there and file issues under <https://github.com/unitsofmeasurement/uom-se/issues>. Or submit those Pull-requests there. Unlike Unit-RI which is only for EG members, we can merge appropriate changes there more easily from non EG or JCP members.  We'll keep the story open as we probably add something to the RI user guide regarding precision and why we left BigDecimal out, but for the RI there is no immediate action item. I hope you understand the rationale and have no problem using Java SE 8 or above? As soon as Eclipse IP helps us to get JSR 363 into its Orbit repository, we can also work on Eclipse UOMo again which is currently Java SE 6 backward compatible and uses BigDecimal in the same ways mentioned. UOMo ".next" should at least work under Java SE 7 like most Eclipse projects, too. |
| Comment by [lfoppiano](https://java.net/jira/secure/ViewProfile.jspa?name=lfoppiano) [ 17/Mar/16 ] |
| Thanks for your quick answer. I completely understand your remarks and I agree on moving to the Java SE 8 implementation.  Now, as in the past months, the only limitation we had is to switch to Java 8.  We probabaly have to do it at some point so maybe this is a good opportunity.  Regards |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Mar/16 ] |
| That would be good. If you run into serious problems and need Java 7, UOMo shall be updated very soon (I may not have to wait for Eclipse IP confirmation at least in the incubator, later it should all be in Orbit of course) However, what's in uom-se aims at "future JDKs" from Java 8 on, and Oracle has already declared Java 7 as "End of Life" for most cases, so it's probably good to migrate if you can.  I consider what's raised here a viable chapter for the RI User Guide: <https://www.gitbook.com/book/unitsofmeasurement/unit-ri-userguide/details>  It is also in GitHub, and while adding a new page or chapter in the first place should be done by owners and editors of the GitBook, everybody is free to raise PRs for those files, too. So you could participate in these chapters if you like. Unlike the Spec or actual RI, this is also a "collaborative" part of the project open to non JCP or EG members alike. |
| Comment by [lfoppiano](https://java.net/jira/secure/ViewProfile.jspa?name=lfoppiano) [ 18/Mar/16 ] |
| Thanks a lot for your complete answer. I will try to migrate to java 8 (which makes sense since java 7 is anyway 5 years old already) and we'll see what to do if we have problems we cannot solve. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Oct/16 ] |
| Handled in RI guide |

|  |  |
| --- | --- |
| **[[Clarification] best way to convert Product of units to product of baseUnits](https://java.net/jira/browse/UNITSOFMEASUREMENT-182)** ([UNITSOFMEASUREMENT-182](https://java.net/jira/browse/UNITSOFMEASUREMENT-182)) [UNITSOFMEASUREMENT-183] [getProductUnits should probably be renamed](https://java.net/jira/browse/UNITSOFMEASUREMENT-183) Created: 02/Mar/16  Updated: 30/Jun/16  Resolved: 30/Jun/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Duplicate** | | | | | duplicates | [~~UNITSOFMEASUREMENT-88~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-88) | SystemOfUnits need to tell its base u... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-193~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-193) | What about Dimension.getProductDimens... | Resolved | | relates to | [~~UNITSOFMEASUREMENT-182~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-182) | [Clarification] best way to convert P... | Resolved | |  |
| **Tags:** | naming |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Based on PR that as such was closed, but documented: <https://github.com/unitsofmeasurement/unit-ri/pull/33>  It occured, that Unit method getProductUnits() could be misleading. >I look at the issue 33 and indeed I believe that the name getProductUnit is >misleading, it should be getBaseUnits or getBaseUnitDecomposition since it >returns the canonical decomposition of the unit (mapping base unit -> >exponent). Indeed, such mapping exist for base units (single unit mapped >to 1) and should not return null  getBaseUnits() sounds shorter, but if there are other suggestions and arguments please elaborate here. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Mar/16 ] |
| While one issue is related to Unit, the other to SystemOfUnits we may want do consider a method like getBaseUnits() only in one of these. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Mar/16 ] |
| Although until Final there is no guarantee the API doesn't change, I guess we'll duplicate getProductUnits() at least for a certain grace period into getBaseUnits() and deprecate the old form. To clean up implementations or other modules using the old version. Either with 0.9 milestone or at the very least 1.0 it will disappear. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 21/Jun/16 ] |
| I vote for renaming before final; I'd say before the final freeze too just remove the old one. Let's not release something with @deprecated in it already... |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 21/Jun/16 ] |
| Thanks for spotting that issue. I like the getBaseUnitDecomposition() proposal. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 21/Jun/16 ] |
| I guess it's best to do a @deprecated transition at least as of 0.9-SNAPSHOT but since e.g. Parfait and other solutions (those public we should see sooner or later via <https://www.versioneye.com/java/tec.units:unit-ri/references> or similar) start using 0.8 now, I guess even 0.9 intended as "Proposed Final Draft" probably should not have both any more. The Snapshot should, but once it's changed from 0.9-SNAPSHOT to 0.9, I'd say delete the old one. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 21/Jun/16 ] |
| As mentioned in the mail thread where suggestions came up, I still think, that "getBaseUnits" sounds a little less scary than "getBaseUnitDecomposition". JavaDoc states "Returns the base units and their exponent..." the Returns tag also says "the base units and their exponent making up this unit." so while "making up this unit" could somewhat justify the extra "Decomposition" bit, I would prefer getBaseUnits especially because it's **plural** and suggests a Map/collection is returned while getBaseUnitDecomposition sounds more like other single object getters such as getDimension etc. So it feels cleaner from a naming point of view, but if everybody who has an opinion was to prefer it, I could live with both. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 21/Jun/16 ] |
| Also we renamed getCompoundConverters into getConversionSteps which sounds a bit more "friendly" and natural IMHO, so probably stay with a more simple term in case of this task and interface, too? Plus respect the plural as it's also done in UnitConverter. WDYT? |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 21/Jun/16 ] |
| I'm quite neutral on getBaseUnitDecomposition() versus getBaseUnits(). |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-182] [[Clarification] best way to convert Product of units to product of baseUnits](https://java.net/jira/browse/UNITSOFMEASUREMENT-182) Created: 16/Feb/16  Updated: 30/Jun/16  Resolved: 30/Jun/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Major |
| **Reporter:** | [lfoppiano](https://java.net/jira/secure/ViewProfile.jspa?name=lfoppiano) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-183~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-183) | getProductUnits should probably be re... | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-183](https://java.net/jira/browse/UNITSOFMEASUREMENT-183) | [getProductUnits should probably be re...](https://java.net/jira/browse/UNITSOFMEASUREMENT-183) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-193](https://java.net/jira/browse/UNITSOFMEASUREMENT-193) | [What about Dimension.getProductDimens...](https://java.net/jira/browse/UNITSOFMEASUREMENT-193) | Sub-task | Resolved | keilw | |  |
| **Tags:** | clarification, design, documentation, question |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Hi, I'm using unit-ri to get data from text and parse them in order to normalize the quantities back to base unit.  If I have the following unit: 10 g \* km / s and I want to transform it to base unit (kg \* m / s).  I'm currently iterating each element of the product and converting element by element and elevating at the power defined by the dimensions and combining them case by case, but it's kind of a mess.  Do you know if there is an easy way? Any suggestions?  (there is no 'clarification' as type of jira ticket) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Feb/16 ] |
| The exact requirement/acceptance criteria may be a bit specific for a user story, but it seems like a good idea also using those. A few JSRs (mainly 375) did that already in a very early stage. This could be refined and moved to a task under a more general story, but you'll find such tasks here. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-181] [Refine quantities included in API](https://java.net/jira/browse/UNITSOFMEASUREMENT-181) Created: 02/Feb/16  Updated: 16/Mar/16  Resolved: 16/Mar/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-100~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) | Define our criterion for inclusion / ... | Resolved | |  |
| **Tags:** | design, quantity, question |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| TCK showed, we have e.g. ElectricPermittivity in javax.measure.quantity, but it's from Table 4 (<http://www.bipm.org/en/publications/si-brochure/table4.html>) which we won't include in the API. So ElectricPermittivity should go away into si-units or a similar module.  From Table 2 (<http://www.bipm.org/en/publications/si-brochure/section2-2-1.html>) we included e.g.   * MagneticFieldStrength * Luminance but not several others.   Are these 2 critical to the most basic fields of use, or could we stick to just the "top 4"   * Area * Volume * Speed * Acceleration |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-180] [Code Coverage](https://java.net/jira/browse/UNITSOFMEASUREMENT-180) Created: 26/Jan/16  Updated: 02/Jul/16  Resolved: 02/Jul/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | coverage, test, testing |
| **Epic Link:** | [Test](https://java.net/jira/browse/UNITSOFMEASUREMENT-111) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As suggested by LJC we should increase the code coverage of RI and by that where possible API (mostly instantiated through RI, so where feasible those coverage reports should also take "javax.measure" into consideration) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Feb/16 ] |
| Similar to JSR 354 (<https://github.com/JavaMoney/jsr354-api> and <https://github.com/JavaMoney/jsr354-ri>) the build chain should propagate JaCoCo findings to a Coveralls service for JSR 363. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/Feb/16 ] |
| Set up for <https://github.com/unitsofmeasurement/unit-api>. The coverage only includes classes (as other tools like Cobertura also seem to do) so the API project includes just 6 concrete classes. Mostly exceptions are never thrown by API level unit tests, so we should add more tests to increase coverage rather easily. Whether or not other free and open tests also would cover interfaces, we could see, but generally these are primarily tested by the TCK. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Jul/16 ] |
| If RI could reach more than 60% it would be nice, but it's a good result so far (e.g. other JSRs like 354 are at 65% till now) |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-179] [Code formatting](https://java.net/jira/browse/UNITSOFMEASUREMENT-179) Created: 25/Jan/16  Updated: 23/Mar/16  Resolved: 23/Mar/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As reported by LJC, we should improve our code formatting to be more consistent.  What tool/plugin could we use to check this? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 26/Jan/16 ] |
| This looks like a plausible solution:   * <https://github.com/revelc/formatter-maven-plugin> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Mar/16 ] |
| Plugin applied to API and RI |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-178] [Github Website does not link to project resources](https://java.net/jira/browse/UNITSOFMEASUREMENT-178) Created: 24/Jan/16  Updated: 01/May/16  Resolved: 01/May/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Minor |
| **Reporter:** | [kittylyst](https://java.net/jira/secure/ViewProfile.jspa?name=kittylyst) | **Assignee:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The project website does not link to key resources (including this issue tracker). Navigating the project without help is very difficult. The primary github.io website for this project should be refactored to directly point to all available & useful materials. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/May/16 ] |
| As java.net is being shut down and the existence or possible future location of e.g. this issue tracker is completely open, we rather not point to a soon dead link. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-177] [Should Quantity extend Comparable?](https://java.net/jira/browse/UNITSOFMEASUREMENT-177) Created: 30/Dec/15  Updated: 15/Jul/16  Resolved: 15/Jul/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-175~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-175) | AbstractQuantity.compareTo should als... | Resolved | | relates to | [~~UNITSOFMEASUREMENT-167~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-167) | Add Comparable to AbstractUnit | Resolved | |  |
| **Tags:** | design, question, sort, vote |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At the moment AbstractQuantity implements Comparable.  <https://github.com/unitsofmeasurement/unit-ri/blob/master/src/test/java/tec/units/ri/function/QuantitySortTest.java> shows, at least for the RI sorting a list of quantities may therefore require them to be expressed as AbstractQuantity. Similar to JSR 354 (MonetaryAmount) we could also have Quantity extend Comparable. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 11/Jul/16 ] |
| I don't like the idea of implementing Comparable, but I'd leave the AbstractQuantity in the RI implementing it.  Reason being that the interface is broad enough to for it being comparable.  Just my opinion, just not see anything in favor. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/Jul/16 ] |
| Thanks for the input. I agree. Let's see, if there are any strong preferences otherwise. JSR 354 did it on both, but even java.util.Currency does not implement Comparable and doing so with CurrencyUnit means, the theoretical option of using an Enum for currencies in a particular implementations won't work. The reason seems an inspiration by <http://www.joda.org/joda-money/apidocs/org/joda/money/CurrencyUnit.html>. That's a concrete class, one of JodaMoney's weaknesses, but since we adopted one or two names for the JSR 354 API that usage of Comparable must have "creeped" into the API. The same "Joda Stephen" or Roger at Oracle did however only apply it to concrete classes in JSR 310 so let's do so here, too |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 15/Jul/16 ] |
| I would also be in favor of not extending Comparable and let this choice to implementations. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/Jul/16 ] |
| Will leave this to RI or other implementations then. |

|  |  |
| --- | --- |
| **[Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)** ([UNITSOFMEASUREMENT-101](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)) [UNITSOFMEASUREMENT-176] [Consider dropping SPI profile from TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-176) Created: 28/Dec/15  Updated: 03/Apr/16  Resolved: 03/Apr/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-101~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-101) | Create TCK | Resolved | |  |
| **Tags:** | profile, test, testing |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| It seems, the "spi" profile for the TCK has more or less the same effect as "full". So we might consider dropping it for the sake of simplicity. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Apr/16 ] |
| Actually SPI covers "core", "format" and "spi" groups, but does not require a full or partial set of quantities. Therefore it makes sense to keep both. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-175] [AbstractQuantity.compareTo should also compare other Unit](https://java.net/jira/browse/UNITSOFMEASUREMENT-175) Created: 23/Dec/15  Updated: 30/Dec/15  Resolved: 30/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-167~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-167) | Add Comparable to AbstractUnit | Resolved | | relates to | [~~UNITSOFMEASUREMENT-177~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-177) | Should Quantity extend Comparable? | Resolved | |  |
| **Tags:** | conversion |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| AbstractQuantity.compareTo() takes the value into consideration, but ignores unit of other Quantity.  Instead of  Unit<Q> unit = getUnit();  It should probably just use  Unit<Q> unit = that.getUnit();  So the unit-based doubleValue() takes the other unit into consideration. If units don't match, we may have to check for ArithmeticException or allow throwing it here. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/15 ] |
| Since Java ME 8 Embedded supports Comparator we may simply use the NaturalOrder class like in uom-se. |

|  |  |
| --- | --- |
| **[Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)** ([UNITSOFMEASUREMENT-101](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)) [UNITSOFMEASUREMENT-174] [Consider using Reflections.org for TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-174) Created: 21/Dec/15  Updated: 30/Mar/16  Resolved: 08/Feb/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | test | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-101~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-101) | Create TCK | Resolved | |  |
| **Tags:** | test, utils |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Instead of low level reflection some of the TCK could use Reflections.org (<https://github.com/ronmamo/reflections>) especially its ReflectionUtils or similar helper classes. |

|  |  |
| --- | --- |
| **[Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)** ([UNITSOFMEASUREMENT-101](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)) [UNITSOFMEASUREMENT-173] [Verify minimum Java version for TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-173) Created: 11/Dec/15  Updated: 30/Mar/16  Resolved: 22/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-101~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-101) | Create TCK | Resolved | |  |
| **Tags:** | java, jvm, version |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As of now the TCK builds against Java 7. TestNG itself is backward-compatible with even Java 5, so unless there are other components with a minimum version of 7, we should try to also make the TCK run on Java SE 6 as a minimal JVM version. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Dec/15 ] |
| There are aspects of the TCK like Serialization tests dependent on Java 7. While it could be possible to back-port it, running tools against the same (minimum) Java version as runtime does not seem essential. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-172] [Units.KILOMETRES\_PER\_HOUR should be rendered "km/h"](https://java.net/jira/browse/UNITSOFMEASUREMENT-172) Created: 02/Dec/15  Updated: 30/Dec/15  Resolved: 30/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-161~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-161) | Do we also want UnitFormat.alias()? | Resolved | |  |
| **Tags:** | format, ui |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Right now, SimpleUnitFormat labels Units.KILOMETRES\_PER\_HOUR like this  DEFAULT.label(Units.KILOMETRES\_PER\_HOUR, "kph");  Based on <https://en.wikipedia.org/wiki/Kilometres_per_hour#Kilometres_per_hour_as_an_abbreviation> it seems better to use "km/h" as default. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-171] [Clarify JavaDoc of UnitFormat.parse()](https://java.net/jira/browse/UNITSOFMEASUREMENT-171) Created: 25/Nov/15  Updated: 30/Dec/15  Resolved: 30/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | format, parse |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At the moment UnitFormat documents the parse() method as follows: <http://unitsofmeasurement.github.io/unit-api/site/apidocs/javax/measure/format/UnitFormat.html#parse(java.lang.CharSequence>)  > If there is no unit to parse the unitary unit (dimensionless) is returned and >@throws ParserException if any problem occurs while parsing the specified character sequence (e.g. illegal syntax). sound slightly contradictory, i.E. returning null is not explicitely outruled, although it's recommended, another constant like ONE would be returned instead.  At least SimpleUnitFormat currently does return null under certain circumstances (e.g. a malformed or unknown string) while other implemetations (mostly on the Java SE or unit system extensions) like EBNFUnitFormat throw exceptions like those documented in the interface.  UnitFormat should get an improved JavaDoc, probably along the lines of JSR 354 MonetaryAmountFormat.parse() which state, the return value should not be null but throw exceptions instead. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Nov/15 ] |
| It seems primarily parseProductUnit() of SimpleUnitFormat should be modified to behave similar as parseSingleUnit() already does. There a null value or similar unknown conditions trigger a ParseException. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-170] [Serializable support](https://java.net/jira/browse/UNITSOFMEASUREMENT-170) Created: 24/Nov/15  Updated: 11/Mar/16  Resolved: 24/Nov/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [snakesvx](https://java.net/jira/secure/ViewProfile.jspa?name=snakesvx) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-4~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-4) | Getting rid of Serializable | Resolved | |  |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| While trying out the measurements API we noticed that most of the interfaces/classes don't seem to support serialization.  Would it possible to add this support where needed/possible?  This would help a lot in cases where an object that (for example) includes a quantity that needs to be either transmited (rmi, ...) or stored (caches, ...). |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Nov/15 ] |
| Thanks for pointing this out. There's been at least one ticket for the SE port (and proposed possible contribution to an appropriate OpenJDK module for IoT/Java SE embedded) <https://github.com/unitsofmeasurement/uom-se/issues/98> following an already solved ticket.  On the API level we won't and can't address that, because Java ME Embedded does not know Serializable. Looking at comparable (value-related) JSRs like 354 or 310, despite some of them aiming only at Java SE, their API or API-like interfaces, e.g. MonetaryAmount or Temporal do not extend Serializable either, but implementing classes like Money or LocalDateTime do. For Java SE we follow the same pattern. If you face problems with serialization not yet addressed in uom-se, please don't hesitate to file further GitHub tickets for that project. On an API level I'm closing this as it's out of scope. |
| Comment by [ceefour](https://java.net/jira/secure/ViewProfile.jspa?name=ceefour) [ 10/Mar/16 ] |
| +1 for making SE port / RI have Serializable support.  If the APIs can't have Serializable well I guess we have to live with it, but please make abstracts Serializable. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/Mar/16 ] |
| For the SE port, see <https://github.com/unitsofmeasurement/uom-se/issues/98> |

|  |  |
| --- | --- |
| **[Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)** ([UNITSOFMEASUREMENT-101](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)) [UNITSOFMEASUREMENT-169] [Consider additional profile(s) for TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-169) Created: 24/Nov/15  Updated: 30/Mar/16  Resolved: 28/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | test | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-101~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-101) | Create TCK | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-129~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-129) | JDK6 compatibility for unit-ri | Resolved | |  |
| **Tags:** | profile, test |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Given at least some Embedded solutions in need of JSR 363 expressed they have dependencies as far back as Java 5, not just 6, as well as being more flexible on smaller devices, we should consider adding another profile or two.  "spi" includes the Bootstrap class which uses Java ServiceLoader (also on ME 8 Embedded a standard, hence it's available even on smaller ME devices) a feature introduced with Java 6. Thus, if a Java 5 "backport" of the RI wants to use the API without changing or porting it, too, all other packages except "spi" seem compatible with that. Hence combining "format" and "quantity" (if also doing so for "base\_quantity" is necessary, please share your thoughts) into at least one more profile sounds like a good idea. Chaining profiles similar to say Maven would be an alternative, but it would complicate calling the TCK, so while groups inside a profile are flexible, I'd rather keep the selected profile as one (also similar to e.g. Java SE or EE) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Nov/15 ] |
| The issue also refers to Java 5/6 compatibility. Especially Java 5 won't support SPI/ServiceLoader, therefore a combination of "format" and other optional packages than "spi" makes sense for such environments. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-168] [Shall we add more Angle units to Units class?](https://java.net/jira/browse/UNITSOFMEASUREMENT-168) Created: 15/Nov/15  Updated: 23/Nov/15  Resolved: 23/Nov/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, question, vote |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| RI Units class contains the SI standard type RADIAN, while the SE port also provides DEGREE\_ANGLE, MINUTE\_ANGLE and SECOND\_ANGLE.  These are not mandatory SI units, but "accepted for use with SI units". If it's not too much overhead and useful in an embedded context, we may add them to the RI, too. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 23/Nov/15 ] |
| I think we already defined what's to be included, no? I think we don't need to add more as it would contradict that definition. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Nov/15 ] |
| We may leave it in the SE port (or think about other places like uom-systems), so for RI, let's leave them out. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Nov/15 ] |
| Stick to the only standard unit in RI for footprint reasons. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-167] [Add Comparable to AbstractUnit](https://java.net/jira/browse/UNITSOFMEASUREMENT-167) Created: 15/Nov/15  Updated: 19/Mar/16  Resolved: 19/Mar/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-177~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-177) | Should Quantity extend Comparable? | Resolved | | relates to | [~~UNITSOFMEASUREMENT-175~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-175) | AbstractQuantity.compareTo should als... | Resolved | |  |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Unlike the SE port, AbstractUnit in the RI does not implement Comparable. As it is available in CLDC8, it should be added there, too. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Mar/16 ] |
| As Unit can often be implemented by enums, the interface must not extend Comparable, otherwise it would prevent using it by Enum types. For Quantity we normally don't expect it to be implemented by enums. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-166] [Multiples of units need to be explicitely labeled, too](https://java.net/jira/browse/UNITSOFMEASUREMENT-166) Created: 20/Oct/15  Updated: 28/Sep/16 | |
| **Status:** | Open |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [1.x](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17750) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Duplicate** | | | | | is duplicated by | [~~UNITSOFMEASUREMENT-143~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-143) | Multiples of HERTZ wrongly formatted | Resolved | |  |
| **Tags:** | format, ui |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At the moment, UnitFormat.label() helps format units in most cases where the symbol may not be available or a different display character is required for a particular unit.  As examples like <https://github.com/unitsofmeasurement/uom-demos/blob/master/domain/energy/src/main/java/tec/uom/demo/energy/DemoUnitSystem.java> show, this right now has to be done for all known (or used) multiples and submultiples of a unit, too. E.g.  SimpleUnitFormat.getInstance().label(WATTHOUR, "Wh");  SimpleUnitFormat.getInstance().label(KILO(WATTHOUR), "KWh");  Even with SimpleUnitFormat known multiples and sub-multiples (like those defined by MetricPrefix of a unit should ideally be formatted automatically, so "Wh" becomes "KWh", "MWh", "mWh" etc. when KILO, MEGA or MILLI are applied. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-165] [Add getSystemUnit() method to QuantityFactory](https://java.net/jira/browse/UNITSOFMEASUREMENT-165) Created: 27/Sep/15  Updated: 27/Sep/15  Resolved: 27/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | api, design, factory |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| QuantityFactory should expose the method getSystemUnit() similar to Unit. |

|  |  |
| --- | --- |
| **[Resolve License issue with Eclipse](https://java.net/jira/browse/UNITSOFMEASUREMENT-156)** ([UNITSOFMEASUREMENT-156](https://java.net/jira/browse/UNITSOFMEASUREMENT-156)) [UNITSOFMEASUREMENT-164] [Change License headers in API](https://java.net/jira/browse/UNITSOFMEASUREMENT-164) Created: 24/Sep/15  Updated: 10/Oct/15  Resolved: 10/Oct/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-163~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-163) | Change LICENSE.txt in API Repo | Resolved | |  |
| **Tags:** | license |
| **Sprint:** | Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This is of lower priority, since every license header refers to LICENCE.txt anyway, but if the license in the API repo was changed, we could simply unify all licence headers and quote the full (BSD) license there, too. |

|  |  |
| --- | --- |
| **[Resolve License issue with Eclipse](https://java.net/jira/browse/UNITSOFMEASUREMENT-156)** ([UNITSOFMEASUREMENT-156](https://java.net/jira/browse/UNITSOFMEASUREMENT-156)) [UNITSOFMEASUREMENT-163] [Change LICENSE.txt in API Repo](https://java.net/jira/browse/UNITSOFMEASUREMENT-163) Created: 23/Sep/15  Updated: 08/Oct/15  Due: 23/Oct/15  Resolved: 08/Oct/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | unplanned | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-164~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-164) | Change License headers in API | Resolved | |  |
| **Tags:** | license |
| **Sprint:** | Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Will first confirm with Eclipse reps, and others, that they're happy and see no reason against adding JSR 363 into public repositories like (Eclipse) Orbit if this was done. |

|  |  |
| --- | --- |
| **[Resolve License issue with Eclipse](https://java.net/jira/browse/UNITSOFMEASUREMENT-156)** ([UNITSOFMEASUREMENT-156](https://java.net/jira/browse/UNITSOFMEASUREMENT-156)) [UNITSOFMEASUREMENT-162] [Put full Spec License into Specification Document](https://java.net/jira/browse/UNITSOFMEASUREMENT-162) Created: 23/Sep/15  Updated: 23/Sep/15  Resolved: 23/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | Maintenance Release |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-161] [Do we also want UnitFormat.alias()?](https://java.net/jira/browse/UNITSOFMEASUREMENT-161) Created: 14/Sep/15  Updated: 03/Dec/15  Resolved: 03/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Incomplete | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-78~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-78) | Can we improve Formatting by adding s... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-124~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-124) | Introduce a Unit "DB" | Resolved | | relates to | [~~UNITSOFMEASUREMENT-172~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-172) | Units.KILOMETRES\_PER\_HOUR should be r... | Resolved | |  |
| **Tags:** | format, i18n/l10n, question, vote |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Public Draft, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Right now, UnitFormat got a label() method similar to those in JSR-275 or 108.  UnitFormat.getLocaleInstance().alias(METRE.multiply(0.3048), "foot");  UnitFormat.getLocaleInstance().alias(METRE.multiply(0.3048), "feet");  UnitFormat.getLocaleInstance().alias(METRE, "meter");  UnitFormat.getLocaleInstance().alias(METRE, "metre");  To help formatting beyond resources or where those are not available (especially locale and resource-neutral environments). While label() addresses formatting and output, the alias() method as above examples show, also can be used for culture- or locale-specific, semantic interpretation, e.g. plural. However, simply knowing METRE.multiply(0.3048) can be both foot and feet does not answer the question of pluralization. That's also in a more extensive, but Locale sensitive way addressed by CLDR data. So while label() is a useful hint to most UnitFormat implementations, it is not entirely clear, if adding alias() without locale or similar semantic meta-information would be beneficial. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 26/Oct/15 ] |
| While I see this being useful on some cases, this is better left to implementations. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 27/Oct/15 ] |
| Leaving to implementations for now. May revisit in a later version if necessary. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Dec/15 ] |
| I want to revisit this method for possible consideration.  A question in one of our forums was about parsing a unit like Units.KILOMETRES\_PER\_HOUR from a string like "km/h".  It seems the established default by SI, EU and other bodies is in fact "km/h", so we should also address this in the RI which uses:  DEFAULT.label(Units.KILOMETRES\_PER\_HOUR, "kph");  as of now (separate ticket).  However, while UnitFormat.label() is meant to change the "default" representation for all instances of that particular format, UnitFormat.alias() would allow to add alternate representations to be used at least by UnitFormat.parse() in addition to the default. So applied properly, both "km/h" and "kph" or "kmph" (was also widely used especially in the UK) could all provide a similar result.  WDYT? |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 03/Dec/15 ] |
| I understand that's useful, but it'll add complication for some use cases only. I stand by my "leave it to implementations" comment as before. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Dec/15 ] |
| Then let's defer it to a later version. I believe some implementations may use it now. Should there be major demand (for alternatives e.g. when parsing) in feedbacks, the later version could be until 1.0, otherwise it's worth keeping it in mind for further versions of the spec/API. JIRA has no "deferred", so I'll abuse "Incomplete", we should certainly not "close" such tickets unlike the ones entirely finished. |

|  |  |
| --- | --- |
| **[Portability of TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-153)** ([UNITSOFMEASUREMENT-153](https://java.net/jira/browse/UNITSOFMEASUREMENT-153)) [UNITSOFMEASUREMENT-160] [Configure Profiles for TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-160) Created: 03/Sep/15  Updated: 04/Sep/15  Resolved: 04/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | portability, profile, test, testing |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to Java EE or ME, the TCK should allow to execute only parts that apply to a certain profile. Given the current optionality, the following profiles or groups look reasonable   * Core * Format * Quantity * SPI (requires at least Format, too)   CDI deals with this via TestNG groups: <http://docs.jboss.org/cdi/tck/reference/latest/en-US/html/configuration.html#tck-properties>  Either doing that or in combination with system properties (and where Maven is used likely its profiles) we should be able to configure which tests in the TCK shall be executed for which profile.  Also see <http://stackoverflow.com/questions/7150302/excluding-testng-groups-from-maven> <http://stackoverflow.com/questions/4578971/appending-jvm-parameter-to-the-parameter-specified-explicitly-in-plugin-configur> |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Sep/15 ] |
| At the moment, passing a wrong or unknown profile name (e.g. "minime" instead of "minimal") as system property causes an exception in TCKRunner  -------------------------------------------------------  T E S T S  -------------------------------------------------------  Running tec.units.tck.TCKRunnerTest  Configuring TestNG with: TestNG652Configurator  Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.601 sec <<< FAILURE! - in tec.units.tck.TCKRunnerTest  testTCKRunner(tec.units.tck.TCKRunnerTest) Time elapsed: 0.024 sec <<< FAILURE!  java.lang.IllegalArgumentException: No enum constant tec.units.tck.util.TestGroups.Profile.minime  at java.lang.Enum.valueOf(Enum.java:236)  at tec.units.tck.util.TestGroups$Profile.valueOf(TestGroups.java:72)  at tec.units.tck.TCKRunner.<init>(TCKRunner.java:81)  at tec.units.tck.TCKRunnerTest.testTCKRunner(TCKRunnerTest.java:41)  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  at java.lang.reflect.Method.invoke(Method.java:606)  at org.testng.internal.MethodInvocationHelper.invokeMethod(MethodInvocationHelper.java:84)  at org.testng.internal.Invoker.invokeMethod(Invoker.java:714)  at org.testng.internal.Invoker.invokeTestMethod(Invoker.java:901)  at org.testng.internal.Invoker.invokeTestMethods(Invoker.java:1231)  at org.testng.internal.TestMethodWorker.invokeTestMethods(TestMethodWorker.java:127)  at org.testng.internal.TestMethodWorker.run(TestMethodWorker.java:111)  at org.testng.TestRunner.privateRun(TestRunner.java:767)  at org.testng.TestRunner.run(TestRunner.java:617)  at org.testng.SuiteRunner.runTest(SuiteRunner.java:334)  at org.testng.SuiteRunner.runSequentially(SuiteRunner.java:329)  at org.testng.SuiteRunner.privateRun(SuiteRunner.java:291)  at org.testng.SuiteRunner.run(SuiteRunner.java:240)  at org.testng.SuiteRunnerWorker.runSuite(SuiteRunnerWorker.java:52)  at org.testng.SuiteRunnerWorker.run(SuiteRunnerWorker.java:86)  at org.testng.TestNG.runSuitesSequentially(TestNG.java:1224)  at org.testng.TestNG.runSuitesLocally(TestNG.java:1149)  at org.testng.TestNG.run(TestNG.java:1057)  at org.apache.maven.surefire.testng.TestNGExecutor.run(TestNGExecutor.java:69)  at org.apache.maven.surefire.testng.TestNGDirectoryTestSuite.executeSingleClass(TestNGDirectoryTestSuite.java:120)  at org.apache.maven.surefire.testng.TestNGDirectoryTestSuite.execute(TestNGDirectoryTestSuite.java:104)  at org.apache.maven.surefire.testng.TestNGProvider.invoke(TestNGProvider.java:113)  at org.apache.maven.surefire.booter.ForkedBooter.invokeProviderInSameClassLoader(ForkedBooter.java:200)  at org.apache.maven.surefire.booter.ForkedBooter.runSuitesInProcess(ForkedBooter.java:153)  at org.apache.maven.surefire.booter.ForkedBooter.main(ForkedBooter.java:103)  This seems acceptable, the exception tells what's wrong, or should we rather override it with no profile ("full")? |

|  |  |
| --- | --- |
| **[Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)** ([UNITSOFMEASUREMENT-101](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)) [UNITSOFMEASUREMENT-159] [Move test-audit.xml to main/resources](https://java.net/jira/browse/UNITSOFMEASUREMENT-159) Created: 31/Aug/15  Updated: 31/Aug/15  Resolved: 31/Aug/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-118~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-118) | Add Tests according to Spec Chapter 4 | Resolved | |  |
| **Tags:** | file, layout, test, testing |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to CDI, the test-audit.xml file should move from the project root folder to src/main/resources. |

|  |  |
| --- | --- |
| **[Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)** ([UNITSOFMEASUREMENT-101](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)) [UNITSOFMEASUREMENT-158] [Add Tests according to Spec Chapter 5](https://java.net/jira/browse/UNITSOFMEASUREMENT-158) Created: 31/Aug/15  Updated: 02/Jul/16  Resolved: 02/Jul/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | spec, spi, test, testing |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Add tests for SPI |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-157] [Shall we keep the name MetricPrefix?](https://java.net/jira/browse/UNITSOFMEASUREMENT-157) Created: 26/Aug/15  Updated: 01/Sep/15  Resolved: 01/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-132~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-132) | Could we rename SIPrefix to MetricPre... | Resolved | |  |
| **Tags:** | design, naming, prefix |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The BIPM chapter on prefixes <http://www.bipm.org/en/publications/si-brochure/chapter3.html> explicitely calls the section **SI prefixes**.  SIPrefix was renamed to MetricPrefix mainly to avoid misconception between the RI where these prefixes are offered for convenience and a dedicated \***SI**\* module (<https://github.com/unitsofmeasurement/si-units>) The term "Metric" was also recommended by a few EC members before JSR 275 was stopped, though the vast majority of libraries prefer the term SI at least for the actual unit system.  WDYT? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Sep/15 ] |
| Based on <https://en.wikipedia.org/wiki/Metric_prefix> also mentioned in the class, let's stick to MetricPrefix. While SI standardized it, there are other systems or catalogues like ISQ (related to ISO80k) or UCUM all using the same prefix here, so not tying the name to SI (like the SI module) probably isn't a bad idea. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-156] [Resolve License issue with Eclipse](https://java.net/jira/browse/UNITSOFMEASUREMENT-156) Created: 23/Aug/15  Updated: 07/Mar/17  Resolved: 07/Mar/17 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.x](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17750) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | unplanned | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-162](https://java.net/jira/browse/UNITSOFMEASUREMENT-162) | [Put full Spec License into Specificat...](https://java.net/jira/browse/UNITSOFMEASUREMENT-162) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-163](https://java.net/jira/browse/UNITSOFMEASUREMENT-163) | [Change LICENSE.txt in API Repo](https://java.net/jira/browse/UNITSOFMEASUREMENT-163) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-164](https://java.net/jira/browse/UNITSOFMEASUREMENT-164) | [Change License headers in API](https://java.net/jira/browse/UNITSOFMEASUREMENT-164) | Sub-task | Resolved | keilw | |  |
| **Tags:** | eclipse, external, library, license |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Due to a different understanding of what **API** vs. **Implementation** are by Eclipse Foundation, there's a blocking issue based on the current **LICENSE.txt** file in Unit-API see <https://java.net/jira/browse/UNITSOFMEASUREMENT-64> This Bugzilla ticket lines out the problem for a downstream Eclipse project (SmartHome) wanting to use JSR 363: <https://bugs.eclipse.org/bugs/show_bug.cgi?id=434006>  Note, this is a **blocker** mainly for Eclipse and downstream projects like SmartHome. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Sep/15 ] |
| So based on what was mentioned in a discussion with Jean-Marie, Eclipse and PMO reps, we would propose a change by putting the full Spec License text into the Specification document (<https://docs.google.com/document/d/12KhosAFriGCczBs6gwtJJDfg_QlANT92_lhxUWO2gCY/edit#heading=h.ejfpuk6yf28c>) instead of a short version like JSR 354 did plus putting a LICENSE.txt that matches the BSD 3 clause license here <https://github.com/unitsofmeasurement/unit-api/blob/master/LICENSE.txt> make Eclipse "happy" without violating any mandatory requirements for JSRs from a JCP/PMO point of view? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Oct/15 ] |
| This (based on a note from JSR 375) pretty much captures the understanding by Eclipse and others on spec vs. software:  // This source code implements specifications defined by the Java  // Community Process. In order to remain compliant with the specification  // DO NOT add / change / or delete method signatures! |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 28/Sep/16 ] |
| SmartHome plans another CQ or update of existing ones for 1.0. This may take a while, but not in our control. Will close if the actual CQs are resolved. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Mar/17 ] |
| The CQs for Unit-API <https://dev.eclipse.org/ipzilla/show_bug.cgi?id=12206> And UoM-SE <https://dev.eclipse.org/ipzilla/show_bug.cgi?id=12610> were both approved by Eclipse IP team. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-155] [Create separate RI documentation](https://java.net/jira/browse/UNITSOFMEASUREMENT-155) Created: 28/Jul/15  Updated: 24/Nov/15  Resolved: 03/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-97~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-97) | Add a chapter about performance | Resolved | |  |
| **Tags:** | documentation, implementation, tutorial |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to JSR 354 (<https://github.com/JavaMoney/jsr354-ri/blob/master/src/main/asciidoc/userguide.adoc>) we may want to create a separate RI guide in the most convenient form   * Asciidoc (lik 354) * Google Docs (like the Spec or also used by other JSRs e.g. 107) * Something else (GitBook was mentioned, haven't used it so far)   Please suggest and feel free to help if we come to a conclusion regarding the best format.  As soon as a "working skeleton" for this new document exists, those willing to help with it can copy over relevant parts of the Spec Document (mostly those under "Implementation") and we may factor them out of the spec.  A bit similar to JSR 354, it does mention RI elements, but some especially those specific to the RI alone are mostly found in the RI guide. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Sep/15 ] |
| Done, the RI User Guide is now here <https://github.com/unitsofmeasurement/unit-ri-userguide> |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-154] [How to tell, if UnitFormat is locale-sensitive?](https://java.net/jira/browse/UNITSOFMEASUREMENT-154) Created: 28/Jul/15  Updated: 20/Sep/15  Resolved: 20/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-78~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-78) | Can we improve Formatting by adding s... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-149~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-149) | Create UnitFormat.format() method wit... | Resolved | |  |
| **Tags:** | format, i18n, i18n/l10n, question, vote |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| While direct references to Locale or ResourceBundle are a no-go in the API or SPI, do we plan to tell users of UnitFormat whether or not the format is locale-agnostic or locale-sensitive?  At the moment, the enum FormatBehavior (<https://github.com/unitsofmeasurement/unit-ri/blob/master/src/main/java/tec/units/ri/format/FormatBehavior.java>) lives on an implementation level only. It is so far just used by QuantityFormat in the RI (not based on an API/SPI level interface, whether we want to change that or not is another question)  There would be several options, one could be adding FormatBehavior or a similar enum to the API (enums are a bit bigger than interfaces, but it would not cause significant growth of the optional "format" package)  It is a bit strange in JSR 107 having an enum just for "STORE\_BY\_REFERENCE" but it seems, the idea is, other optional features may be added there and its only SPI element CachingProvider ( <https://ignite.incubator.apache.org/jcache/1.0.0/javadoc/javax/cache/spi/CachingProvider.html>) allows to check, whether it is supported or not. We could do something similar with an enum like FormatBehavior or as a smaller alternative simply offer a flag like isLocaleSensitive() or similar. Two examples (both IBM products now ) show this, a Tivoli Workflow (<http://www-01.ibm.com/support/knowledgecenter/SS2GNX_7.2.1/com.ibm.tivoli.tpm.soapjdoc.doc/com/ibm/tivoli/orchestrator/de/soap/Workflow.html>) or iLog JViews IlvGraphic (<http://docs.roguewave.com/jviews/8.9/jviews-framework89/doc/html/en-US/refjava/html/ilog/views/IlvGraphic.html>)  Should we plan to support this in the API, then adding something similar to JSR354 AmountFormatQueryBuilder (<https://github.com/JavaMoney/jsr354-api/blob/master/src/main/java/javax/money/format/AmountFormatQueryBuilder.java>) along the lines of QuantityFactory would be a necessary addition to the API. Without passing Locale explicitely, we may need to do this more sensitive, e.g. with a Class reference similar to the way ServiceProvider already does that for types the API does not need to know.  Otherwise it would be implementation-specific, and users who need localized formatting other than the default locale, would have to call methods directly at a class like LocalUnitFormat in UOM-SE: <https://github.com/unitsofmeasurement/uom-se/blob/master/src/main/java/tec/uom/se/format/LocalUnitFormat.java>  WDYT? |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-153] [Portability of TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-153) Created: 27/Jul/15  Updated: 04/Sep/15  Resolved: 04/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-101~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-101) | Create TCK | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-160](https://java.net/jira/browse/UNITSOFMEASUREMENT-160) | [Configure Profiles for TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-160) | Sub-task | Resolved | keilw | |  |
| **Tags:** | embedded, portability, test, testing |
| **Epic Link:** | [Test](https://java.net/jira/browse/UNITSOFMEASUREMENT-111) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At least some utility classes in the initial TCK codebase use BigDecimal. If possible this should be avoided, given the API also does not use types that would be incompatible with target platforms like Java ME. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Sep/15 ] |
| TestUtils was modified to avoid BigDecimal or other elements of java.math. However, since the TCK itself is not meant to run on Java ME Embedded, there are crucial parts including TCKRunner dependent on **reflection**. These dependencies are OK since the TCK won't have to run on an Embedded VM, it just has to test artifacts that do in a platform-neutral way. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-152] [Refactoring SPI](https://java.net/jira/browse/UNITSOFMEASUREMENT-152) Created: 27/Jul/15  Updated: 15/Jun/16  Resolved: 15/Jun/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-145~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-145) | Move Bootstrap mechanism into API | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-121~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) | Introduce "Pluggable" Unit Systems an... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-189~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-189) | Should "Bootstrap" be final or abstract? | Resolved | | relates to | [~~UNITSOFMEASUREMENT-186~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-186) | Change ServiceProvider.available() to... | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-186](https://java.net/jira/browse/UNITSOFMEASUREMENT-186) | [Change ServiceProvider.available() to...](https://java.net/jira/browse/UNITSOFMEASUREMENT-186) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-189](https://java.net/jira/browse/UNITSOFMEASUREMENT-189) | [Should "Bootstrap" be final or abstract?](https://java.net/jira/browse/UNITSOFMEASUREMENT-189) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-190](https://java.net/jira/browse/UNITSOFMEASUREMENT-190) | [Create PoC for a refactored ServicePr...](https://java.net/jira/browse/UNITSOFMEASUREMENT-190) | Sub-task | Resolved | desruisseaux | | [UNITSOFMEASUREMENT-191](https://java.net/jira/browse/UNITSOFMEASUREMENT-191) | [Could QuantityFactoryService be merge...](https://java.net/jira/browse/UNITSOFMEASUREMENT-191) | Sub-task | Resolved | keilw | |  |
| **Tags:** | question, vote |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| A task to clarify, if we're happy with the Bootstrap class as JSR 354 also defined it?  An important difference is, that 354 added several API level facade classes like Monetary or MonetaryRounding which we probably would like to avoid here with Embedded in mind. Where we have similar needs, the RI or other implementations would usually meet that, see Units or Quantities.  Good examples are JSRs from 107 to 349 (Bean Validation) or 365 (CDI 2)  They all have such API level central accessor classes. Named Caching, Validation or CDI. The latter in its "spi" package, simlar to our place for such a class. (since we cannot mandate it for devices without ServiceLoader or Liblet support)  Please suggest names, I don't want to propose too much in this case. The only alternative to Bootstrap I could think of was a CDI inspired acronym like UOM (or UoM. Simply calling it Measure or Measuring (along the lines of JSR 107) were other options, but especially Measure would cause confusion with a similar class from JSR 275, so it is not my favorite  Thanks, Werner |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 27/Jul/15 ] |
| Also see JPA Bootstrap class <http://docs.oracle.com/javaee/7/api/javax/persistence/Persistence.html> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 30/Sep/15 ] |
| Meeting @desruisseaux at ApacheCon, it seems, with pretty much the same lines of code, calling ServiceLoader directly offers a similar value, so we seemingly won't need Bootstrap. See <https://github.com/unitsofmeasurement/uom-demos/blob/master/console/se/src/main/java/tec/uom/demo/se/FactoryDemo.java> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 05/Oct/15 ] |
| See <https://java.net/projects/unitsofmeasurement/lists/experts/archive/2015-10/message/0> other JSRs from 107 to 354 or 365 (CDI 2, already had this kind of "bootstrap" class ever since 1.1) give strong evidence, this kind of class adds value to the JSR, especially by making it more independent of either Java class loading deficits or completely different service loading mechanisms along the lines of OSGi.  So the goal is to decide, if Bootstrap was a proper name for the class or we should find another name. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Mar/16 ] |
| Created a Doodle poll <http://doodle.com/poll/rgwqcdz6u9c3rp7pgue9d9mw/admin#table> (it should only be open to EG members) |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 17/Mar/16 ] |
| An alternative could be:   1. Make ServiceProvider an abstract class instead than an interface (this is not a problem for java.util.ServiceLoader). 2. Move the Bootstrap methods as static methods in ServiceProvider. 3. Remove the Bootstrap class.   This would simplify a little bit the API (one less method) and avoid the difficulty to find a name for Bootstrap. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Mar/16 ] |
| Try if you want in a branch: <https://github.com/unitsofmeasurement/unit-api/tree/simplify_bootstrap>  Aside from the Spec document and other examples, Bootstrap is referred to by many places like tests.  Both JSR 354 and CDI use one or multiple interfaces, only the accessor singleton (like Bootstrap) is an abstract class offering to replace SPI providers. Give it a try. While they do not offer the same level of runtime pluggability, either java.util.Currency or the unit type of ICU4J are abstract classes with some static helpers and factories like "getAll" or "getAvailable...". |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 19/Mar/16 ] |
| Thanks. Just one question: Bootstrap looks for a single instance of ServiceProvider, which in turn returns a List of each services. Isn't ServiceProvider expected to be provided by Unit API implementation (in which case I would expect a list of ServiceProvider, and the service provider returning only one instance of each service), or do we expect implementations to rely on DefaultServiceProvider on not provide their own ServiceProvider (which would explain the current API)? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Mar/16 ] |
| DefaultServiceProvider is just a fallback. You need different implementations and modules (e.g. SI, UCUM or similar) on top, but tests we got there so far show, they're all available and provided. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 19/Mar/16 ] |
| I think that my question was not clear. Current Bootstrap implementation searches for the first ServiceProvider found on the classpath, ignoring all other instances (the ServiceProvider.getPriority() method is apparently ignored). Then ServiceProvider has the following method:   * <T> List<T> getServices(Class<T> serviceType);   My comment was: if ServiceProvider is expected to be provided by Unit implementations, then:   * I don't see why ServiceProvider.getServices(Class) should return a list. Why a given implementation would want to expose many implementations of SystemOfUnitsService or QuantityFactoryService? * The fact that more than one implementation may exist on the classpath would have to be handled in Bootstrap. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Mar/16 ] |
| The inspiration of Bootstrap (and its name) was based on JSR354: <https://github.com/JavaMoney/jsr354-api-bp/blob/master/src/main/java/javax/money/spi/Bootstrap.java>  You'll notice a somewhat strange behavior, that it sorts only by <b>name</b> of services there which is something we removed earlier.  It did not use the priority there either.  PriorityAwareServiceProvider (also in UoM-SE: <https://github.com/unitsofmeasurement/uom-se/blob/master/src/main/java/tec/uom/se/internal/PriorityAwareServiceProvider.java>) brought the notion of Priority via the Common Annotations API ({{ @ Priority}}) and also uses it. getPriority() of the actual provider is something to look into using. ServiceProvider now extends Comparable and Bootstrap (only in master) sorts by priority when selecting the default.  As for a list of services in a modular setting, it's crucial, that multiple modules can provide the same service. e.g. <https://github.com/unitsofmeasurement/si-units> and several modules under <https://github.com/unitsofmeasurement/uom-systems>. The SystemOfUnitsService test in ISO80000 <https://github.com/unitsofmeasurement/uom-systems/blob/master/iso80k/src/test/java/systems/uom/iso80k/internal/SystemOfUnitsServiceTest.java> shows this rather well.  JSR 107 while there is just one provider type CachingProvider also returns a collection of different providers in its singleton: <http://ignite.apache.org/jcache/1.0.0/javadoc/javax/cache/Caching.html> (the equivalent to Bootstrap in the JSR 363 SPI) |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 21/Mar/16 ] |
| I do not question that we can have many service providers in a modular environment. What I find surprising is where this multiplicity is handled: in the ServiceProvider class which, in my understanding of your reply above, is provided by each Unit API implementors. If ServiceProvider is provided by an implementor, then I would expect it to care only about the implementations of that particular implementors (so no List in that interface). If we want to have a different Unit API implementation, then I would expect to have a different ServiceProvider (so a List in Bootstrap instead than in ServiceProvider). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 21/Mar/16 ] |
| There is only one ServiceProvider per implementation. So e.g. the RI has one, while UOM-SE comes with another (currently more powerful) one, but there is no concept of mixing RI and SE port in the same JVM runtime (other JSRs like JSF face pretty much the same constraints) So Bootstrap allows the flexibility to switch to another ServiceProvider. In theory an implementation may offer multiple service providers, e.g. one using Java ServiceLoader and another one using OSGi or in future maybe Jigsaw, but not one ServiceProvider for each service. Therefore the only active ServiceProvider at a given time has to offer a list of services if multiple modules like SI, ISO, UCUM, etc. add multiple services. I'm not totally against the idea of merging Bootstrap and ServiceProvider but all "modern" and even a few "refurbished" (like 107 which started even before 108 ) JSRs that went final in the last few years do not put everything in one abstract "uberclass". While APIs defined much earlier like ICU4J or the now pretty defunct JSR 256 (Sensor API) did. They both combine Unit with static factory methods for getting or even defining new units (both of them have no strict quantity type checking at compile time) which we got SystemOfUnits for.  CDI <https://docs.jboss.org/cdi/api/1.1/javax/enterprise/inject/spi/CDI.html> while its current() method returns an instance of itself still exposes a BeanManager where most of the somewhat comparable methods like getBeans() reside, but it also allows static re-configuration via setCDIProvider() and although multiple providers could be found, only one is the currently used one. Almost every SPI element in CDI requires sub-classing the CDI class to access protected methods like setCDIProvider(). I am not sure, if we gained value from doing that, or it would be more complicated to implement and use. So Bootstrap.getServices() is a convenience method for getServiceProvider().getServices() and same with getService(). They could be removed from Bootstrap if you always wanted to go via the getServiceProvider() but allowing multiple modules to be loaded on top of an implementation you need them in ServiceProvider. Again, the list seems more necessary, the single result method is the default where more than one service of the same type is defined in one module. Very common for formatting services, less common but also applied to others like SystemOfUnitsService. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 22/Mar/16 ] |
| I would like to clarify the purpose of each class. Would it be possible to answer by "Yes" or "No" to the following questions please?   1. Each Unit API implementation provides its own ServiceProvider. For example there is one ServiceProvider provided by RI and another ServiceProvider provided by UOM-SE. 2. If we have many Unit API implementations on the classpath, then we have many ServiceProvider instances: one for each implementation. 3. If I want RI services, I use the ServiceProvider provided by RI implementation. If I want UOM-SE services, I use the ServiceProvider implementation provided by UOM-SE implementation. 4. The ServiceProvider provided by RI implementation does not know about UOM-SE services. Conversely the ServiceProvider provided by UOM-SE implementation does not know about RI services.   If the answer is "yes" to all the above questions, then it seems to me that what we need is List of ServiceProvider, not ServiceProvider methods returning List. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Mar/16 ] |
| Why would we need a list of ServiceProvider implementations?  In practice, simply because they build on multiple incompatible JVMs (JavaME 8 vs. Java SE 8 or maybe SE 6/7 for other implementations) it is impossible right now to use unit-ri and uom-se in at least the same ClassLoader. Having "Bootstrap" (or whatever name we feel was best in the end?) return a list of multiple ServiceProvider might be an option, see <https://docs.jboss.org/cdi/api/1.1/javax/enterprise/inject/spi/CDI.html>, but CDI also returns the "default" one. Which matches our getServiceProvider() method. The one active, configured and in use.  Thus if you have Bootstrap.getServiceProvider() you get the "current" ServiceProvider which is per implementation. In almost every case especially on ME (on Java SE we might see more than one, same for CDI, the enterprise features may not even make their way into the SE profile of CDI 2 or at least return just one CDIProvider) there is just one implementation. Hence ServiceProvider which acts as a stub for either JDK ServiceLoader or alternatives like OSGi must return a List of services. There is no need of an overhead of forcing each module to create something like SISystemServiceProvider because ServiceProvider is incapable of providing more than one service of type SystemOfUnitsService for each module plue UCUMPrintFormatServiceProvider, ImperialSystemServiceProvider vs. USSystemServiceProvider etc. you get the idea.  Offering a list of service providers similar to CDI could be an option, but it is a possible add-on and should only be for power-users. Ideally similar to the way CDI does it, having to sub-class an abstract base class to use it, but I would not even see that as a prio for V1, it is hardly the case that you mix implementations. Also for most other JSRs. Servlet or JSP make it possible to swap a WAR between 2 or more app servers, but nobody would seriously run Glassfish, Tomcat or TomEE in the same JVM at the same time |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 22/Mar/16 ] |
| I have no problem with Bootstrap.getServiceProvider() returning a default provider. What I'm trying to clarify is why the List is in the ServiceProvider interface. To repeat my previous questions: do we agree that there is one ServiceProvider per implementation, and that each ServiceProvider instance do not need to know about other implementations? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Mar/16 ] |
| I'd say there should be one ServiceProvider per "kind of service retrieval". As the RI and SE port use java.util.ServiceLoader by default unless you re-configure it in Bootstrap that way of service-retrieval speaks for one ServiceProvider per implementation. In the rare event that somebody introduced RI and SE port into the same class-loader and an application calls either ServiceProvider it would get all matching services via java.util.ServiceLoader. Even some that may not work in that environment, but e.g. carefully separating calls to RI and SE components without trying a direct cast would work under SE, but not under ME where any call to the SE service causes an immediate failure.  There could and will be other implementations of ServiceProvider e.g. for OSGi (as soon as Eclipse was finally able to resolve <https://java.net/jira/browse/UNITSOFMEASUREMENT-156> on its end, then UOMo may implement JSR 363) or probably in future versions of the SE port an optimized ServiceProvider calling the Jigsaw module system directly instead of ServiceLoader. That is why IMHO I would prefer to keep the singleton (or even abstract) facade "Bootstrap" and ServiceProvider separate, though in theory one could also have more than one sub-class of such abstract singleton/provider, but it feels cleaner and has almost no real overhead in JAR size by 2 SPI elements instead of 1. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Mar/16 ] |
| Although many implementations should be fine with just one ServiceProvider adding Comparable to it now so that getPriority makes sense looks like a logical step. So if an implementation offers one ServiceProvider using ServiceLoader and another one e.g. via OSGi or another service concept, it can give the highest priority to the one it wishes to use. If there were multiple implementations, an implementation compatible with others could also give a higher priority to its default/only ServiceProvider so it gets picked by Bootstrap without having to override it. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 24/Mar/16 ] |
| I agree with desruisseaux's comment with 4 questions. I think we have yes for all questions and we wouldn't need a list.  And the name Bootstrap is fine, but I don't think this is about the name anymore. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Mar/16 ] |
| Thanks for voting. "Something else" suggests you like a different name, but if it's about shifting methods like discussed here' that would also be OK, but someone must make a proposal either in their own Github space or a branch. After I fixed the formatting issue, I deleted the branch which would have too many conflicts, but anybody should be able to create a branch or fork it in their own GitHub repo and offer a PR.  Voting "something else" does not state you are fine with keeping the name Bootstrap that would be option 1 (there are not too many and if someone voted on all it would still get to a preference which is currently "Something else".  I hope we agree on the flexible approach to setting a new "provider" since it is crucial to making the API flexible yet small. Relying entirely on java.util.ServiceLoader would lock out OSGi and cause fragmentation. The TCK might work, but say an implementation needs dynamic service retrieval via something like OSGi, the API will not offer any of that and make different implementors use proprietary solutions. Not to mention that whatever Jigsaw offers in terms of access to the module system, our API will not support that unless some sort of backward-compatible "bridge" was offered via the only solution java.util.ServiceLoader. IMHO I don't care if it's called Bootstrap, UoM or ServiceProvider as long as the flexibility say CDI offers in its abstract accessor class is not lost. If that was a protected or public methods, those are details, first everyone who voted against sticking to Bootstrap must make a working proposal. That works across all the existing modules without breaking its Unit tests each of them has for service retrieval. If Bootstrap got another name or methods were removed (adding them won't break anything) then of course every module must be adjusted to prove the alternativ works. As well as the TCK which must run with all existing tests (and feel free to add more, but that "singleton" is currently key to how the TCK works in a platform neutral way.  Which "4 questions" by @desruisseaux do you mean? There is a post with 3 bullet points suggesting to merge Bootstrap and ServiceProvider into a single abstract base class. As long as it offers some flexibility like CDI, even if it was only accessible to sub-classes if you want to swap the default provider, I am not against that approach. Making it work and demonstrating, all modules (some with more than one service like USSystemService vs. ImperialSystemService, if we restricted that to force a new ServiceProvider per module, not just imlementation, make it work and it could also be OK. A bit more complex, we'd move into CDI-like terrain, but if that's what a majority wants, why not.  I'll leave the doodle open till after the Easter weekend. So some might vote if they are away. Then we have to see what the preference is and if it is "refactor", please do refactor. I could also do such thing, but the ones who feel an alternative was better for the API should present a PoC of this idea.  JSRs like 354 or 310 have also undergone quite a few changes even after a Public Draft. If you (especially all Spec Leads please) believe, doing another Public Draft Review with a changed API was better than a Proposed Final Draft, please also say so. Harold/PMO offered both, we should tell them which one we do next. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Mar/16 ] |
| Please also see <https://java.net/jira/browse/UNITSOFMEASUREMENT-189>. "Bootstrap" is in quotes for a good reason, since the ticket is about any approach that involves an abstract class like in CDI or (with the mentioned limitations) JSON-P. To not "spam" this list, please post matching ideas or questions there, too. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 25/Mar/16 ] |
| Hello Werner.   * We agree about allowing different providers, based on java.util.ServiceLoader, OSGi or whatever. * I suggest to defer the question about Bootstrap name until after we clarified what is ServiceProvider contract. * The 4 questions by @desruisseaux were those posted on March 22th (if I'm not misunderstanding @leomrlima intend):   + Do we agree that each Unit API implementation provides its own ServiceProvider?   + Do we agree that if we have many Unit API implementations on the classpath, then we have many ServiceProvider instances?   + Do we agree that if I want RI services I use the ServiceProvider provided by RI implementation, and if I want UOM-SE services I use the ServiceProvider implementation provided by UOM-SE implementation?   + Do we agree that the ServiceProvider provided by RI implementation does not know about UOM-SE services, and that conversely the ServiceProvider provided by UOM-SE implementation does not know about RI services? * The proposal is to simplify ServiceProvider rather than making it more complex by moving in CDI land. * I will try to do a working proposal in a branch during the next week. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Mar/16 ] |
| Hello Martin/all,  Thanks for pulling those out from the thread. There were other (3) numbered items earlier, so these did not stick out so much.  Glad everyone agrees on supporting different module or service infrastructures. It's not just OSGi or Jigsaw, there have been questions about Android support which has yet another concept of services and every few weeks we see a new "Microservices Framework" by vendor XYZ aiming to be the "ultimate solution" and "Java EE killer"  We also agreed that the RI must work under Java ME 8 as a "minimal platform" and made it work around JavaOne last year. That must remain possible. So any approach in a branch, please either test it in the Java ME 8 SDK or ask others (especially @leomrlima or myself used it on several occasion) to test them. This means, the RI is restricted to "small" numeric types and e.g. BigDecimal for very fine-grained numeric precision is not available. Members of the community already pointed that out and I explained why we decided for leaving BD out of the RI. As the SE port anomg other possible implementations (JScience.next, UOMo,...) offers BigDecimal support they just have to pick the implementation of their choice.  You actually do see multiple implementations of ServiceProvider in the same classpath, e.g. Eclipse discovers multiple implementations when you run apps or tests in a workspace that has both the RI and other implementations loaded. Under Java ME this will simply fail, any non-compliant code fails to execute in the emulator or device. Under Java SE one can have both the RI and other implementations in the same classpath, but mixing them e.g. trying to convert tec.units.ri.unit.BaseUnit to tec.uom.se.unit.BaseUnit will simply fail with a ClassCastException. Since we aim at Java ME 8 as well as SE 7 or 8 and above (e.g. for the whole Big Data space or other desktop and server solutions) multiple implementations are inevitable and highly encouraged. Look at JCache: <https://jcp.org/aboutJava/communityprocess/implementations/jsr107/index.html>. It has at least half a dozen implementations, but do you really think mixing Ehcache, Coherence and Hazelcast in the same classpath and single piece of application code will work???  So although you get services defined by the RI if you run 2 or more implementations in the same class loader, at most you might pick one and avoid mixing it with the other. All implementations must be independent, so as you won't see Coherence use Ehcache or the RI of JSR 107, I guess that explains if we need at least one ServiceProvider implementation per JSR implementation. The answer can only be "Yes" as long as the implementation uses the Full Profile of JSR 363 (remember, SPI is optional, so very small implementations may avoid it and e.g. use an enum for units instead, see the POC <https://github.com/unitsofmeasurement/uom-impl-enum>)  Looking forward to your working proposal. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 29/Mar/16 ] |
| I committed a proposal on <https://github.com/desruisseaux/unit-api.git> There is two commits. The first one is only javadoc fixes (see commit message for details). The second commit:   * Change ServiceProvider into an abstract class. * Remove all methods returning a List. * Replace the getService(Class<T> serviceType) method by three methods, one for each type of services. This is not the intend of this class to replace ServiceLoader or dependency injection frameworks. * Provide the following static methods getAvailables(), getDefault(), setDefault(ServiceProvider). Note that getAvailables() returns an array because it was convenient, but I'm neutral on whether it should return an array or an List. The important thing is that we can have an arbitrary amount of ServiceProvider, not an arbitrary amount of SystemOfUnitsService, UnitFormatService and QuantityFactoryService inside a given ServiceProvider. * Delete Bootstrap since its functionalities moved as ServiceProvider static methods. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Mar/16 ] |
| No time to test now, should actually be asleep, but the merge itself (aside from return values we might think about method names if get\* is necessary, see other JSRs like JSON-P) Guess a clone of your fork locally should work.  I don't know yet how the list-less approach will work in all the cases with several services of the same type, e.g. UnitFormatService in a single module. Possible that the service list can handle all cases, but unit tests and other code has to be tweaked significantly to test if everything works. Probably easier locally. Whoever can, please give it a try. Either also forking them (RI, SI and Systems at least the RI based ones) or making local modifications, but please don't push anything to master yet.  If I see this right, getAvailables() doing a clone() might run into an NPE if during initialization the providers array was not set due to an underlying error. That is a reason, why e.g. a List or similar could be more secure if we set it to an empty collection which can be safely cloned and fill it on succesful init. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 29/Mar/16 ] |
| Indeed, this patch assumes that each implementation would provide at most one UnitFormatService. If an implementation really wants to provide more than one UnitFormatService)} (but I would be curious to know why), it can provide more than one {{ServiceProvider.  The clone() call in getAvailables() should not throw a NPE, since if the array was null and can not be built, then another exception will have been thrown in the getProviders() private method. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Mar/16 ] |
| That assumption is insufficient I'm afraid. See the diagram in <https://github.com/unitsofmeasurement/uom-systems>, we have (for a good reason, each developer and user will need different combinations of standards and systems on top of an implementation) a chain of e.g. RI > SI > ISO80000, or any other system module like Common (US Customary, Imperial) or UCUM.  Therefore you need to handle more than one SystemOfUnitsService (3 in this case) while having just one ServiceProvider implementation is perfectly fine.  As mentioned, the merge as such sounds like a viable option and could simplify the API a little bit (merging 2 elements into one, not many JSRs do this, but we could live with a JSON-P like scenario as long as we allow to swap providers) but modularity and extensibility is absolutely crucial to the success of the JSR. Nobody can or wants to define every possible unit system or quantities in the RI or even other implementations. So unless the ServiceProvider extension based on 3 MANIFEST/services entries across 3 modules could expose all the necessary unit systems through just one service, I'm afraid we need one SystemOfUnitsService per module.  Will take a look at it, please feel free to tweak the fork/patch if you sense it cannot accomplish that. In theory forcing yet another ServiceProvicer per module could work, but I would rather have a more flexible ServiceProvider offer more than one service instead of the overhead of having to declare an unneccessary amount of providers and modules. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 29/Mar/16 ] |
| You can have multiple system of units. The SystemOfUnitsService interface already has getAvailableSystemsOfUnits() and getSystemOfUnits(String) methods for this purpose. No need for lists in ServiceProvider for this use case. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Mar/16 ] |
| That's what I plan to try against existing modules like SI and "uom-systems". |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 30/Mar/16 ] |
| As I feared, it doesn't work for all required cases. Implementing that approach in the RI works quite well: <https://github.com/keilw/unit-ri> All JUnit tests pass.  The "convenience methods" feel a bit redundant as the only way to make service discovery work is via  @Override  public SystemOfUnitsService getSystemOfUnitsService() {  return getService(SystemOfUnitsService.class);  }  We'll need the low level methods, otherwise you're immediately forced to declare yet another ServiceProvider in every module.  UnitFormatService returns 2 formats, because SimpleUnitFormat has 2 flavors "Standard" and "ASCII" both exposed by the service. However, two crucial tests in <https://github.com/keilw/si-units> (fork of the SI extension module) fail.  assertEquals(2, systems.size()); // we'd expect SI and Units here  in SystemOfUnitsServiceTest.testOtherUnitSystems() fails because only 1 SystemOfUnits can be discovered here. Since that's the case, the value of having a getAvailableSystemsOfUnits() method becomes questionable.  The nature of java.util.ServiceLoader is to return an Iterable of services for a given service type class. If you simply take the first occurance, you end up replacing the default (if a new module like SI defines it in META-INF/services but SI can't manipulate or access those previously defined in the RI or vice versa. ServiceProvider.getAvailables() also would return only more than one, if you declare another ServiceProvider on top of the first one in a new module. Having to to through java.util.ServiceLoader every time seems unproblematic, but we'd have to explore it for use cases like OSGi without META-INF/services.  Please feel free to propose a fix for <https://github.com/keilw/si-units> in a new fork. It looks like especially UnitFormatService and SystemOfUnitsService have to be revisited in what we return. We only cripple what at least the JDK ServiceLoader offers if we restrict that to a single item where multiple are offered. Unless somebody offers a good solution to really "merge" those, the only solution could be to have at least SystemOfUnitsService return exactly ONE system per service but be able to return multiple services (the natural offer for every service, even if the collection may just contain a single service) |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 30/Mar/16 ] |
| I do expect every module to provide their own ServiceProvider implementation. Those implementations should not rely on a getService(Class) method, because:   * ServiceProvider.getService(Class) duplicates java.util.ServiceLoader's work in a questionable way: no control on ClassLoader to use and lost deferred instantiation (all implementations of a given service are instantiated even if we do not need them). * ServiceProvider.getService(Class) returns an arbitrary implementation found on the class path. This is probably not what we want. A module providing an Unit implementation probably wants services of the same implementation.   So I expect ServiceProvider to be implemented like below by all modules:  public MyServiceProvider extends ServiceProvider {  @Override  public SystemOfUnitsService getSystemOfUnitsService() {  return new MySystemOfUnitsService(); // NO ServiceLoader usage here.  }  }  I have not yet looked at SystemOfUnitsServiceTest.testOtherUnitSystems(), but I would expect exactly one SystemOfUnitsService implementation for a given module, and the getAvailableSystemsOfUnits() method of that implementation to return 2 system of units. How your SystemOfUnitsService implementation discover those 2 system of units is an implementation detail; only the discovery of the initial ServiceProvider would be done by the META-INF/services mechanism (or OSGi or whatever). You can rely on the META-INF/services mechanism for discovering your SystemOfUnitsService implementation if you want, but you don't have to (I would not, for avoiding unexpected mixing of implementations). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 30/Mar/16 ] |
| Could you give it a try for a fork of <https://github.com/keilw/si-units> then? It's somewhat similar to the more "flat" provider structure of JSR 107 (see <https://github.com/jsr107/RI/tree/master/cache-ri-impl>) or JSON-P. Guess it's doable, but will cause a bit of an overhead for extension modules, e.g. ServiceProvider enforcing 3 "get...Service" methods to be overriden by every module-specific provider, you must always return "something" even if the value might be null or an empty dummy object.  Note, as it was so far, you could simply have module A declare a SystemOfUnitsService and module B declare a UnitFormatService implementation. While QuantityFactoryService is almost exclusively up to the RI or another implementation. We have to question its need in every module. As long as new SI custom types like Action or Density (or any quantity defined by other 3rd party modules) properly extend Quantity there has been no need to create your own QuantityFactory and service in a module like SI to use them. We should not out-rule the possibility of such service, but if the method signature of ServiceProvider forces every module to return something even if they have no need for its own QuantityFactory we have to document that and it will cause redundant methods even if they just returned null.  IMHO for some or all of these, we should provide defaults (either returning null or a safe "stub") in the abstract base class and not force every sub-class to override EVERY method. That might do and still allow each module to extend/override what they really use.  WDYT? |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-151] [Define QuantityFactoryService in API](https://java.net/jira/browse/UNITSOFMEASUREMENT-151) Created: 25/Jul/15  Updated: 14/Sep/15  Resolved: 14/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-145~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-145) | Move Bootstrap mechanism into API | Resolved | |  |
| **Tags:** | factory, service, spi |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The interface QuantityFactoryService currently in SPI part of the RI should ideally be part of the API's SPI, too. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-150] [Move DimensionalModel to "spi" package](https://java.net/jira/browse/UNITSOFMEASUREMENT-150) Created: 23/Jul/15  Updated: 23/Jul/15  Resolved: 23/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, spi |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The DimensionalModel class feels more like an SPI component (probably better in the RI, but if handled via Service loading mechanism there could be use for it in the API/SPI, see CDI and other JSRs) should be moved into "spi" package of the RI. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-149] [Create UnitFormat.format() method with a single Unit argument](https://java.net/jira/browse/UNITSOFMEASUREMENT-149) Created: 13/Jul/15  Updated: 14/Sep/15  Resolved: 14/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-78~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-78) | Can we improve Formatting by adding s... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-154~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-154) | How to tell, if UnitFormat is locale-... | Resolved | |  |
| **Tags:** | format |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Right now UnitFormat defines one format() method with arguments and return values inspired by format(Object obj, StringBuffer toAppendTo, FieldPosition pos) in java.text.Format but without depending on it (ME 8 does not contain java.text)  java.text.Format contains another more simplified method format(Object obj) while other JSRs like 354 or 310 also define similar format(<JSRType>) methods with arguments specific to that JSR, e.g. MonetaryAmount for JSR 354. Both these JSRs also deviate from java.text.Format) as to defining the {{Appendable variant of the format() method as void with no return value. Whether or not we could also do this, maybe subject to a follow-up ticket, let's first try to add the simple method. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-148] [Fix JavaDoc in Units](https://java.net/jira/browse/UNITSOFMEASUREMENT-148) Created: 12/Jul/15  Updated: 09/Nov/15  Resolved: 09/Nov/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | javadoc |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Some constants in the Units system class contain a JavaDoc flaw. Instead of closing </code> it only says /code> resulting in weird/wrong display of the symbols affected. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 28/Jul/15 ] |
| Also seems to exist in other classes. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-147] [Unit.asType() does not throw ClassCastException where required](https://java.net/jira/browse/UNITSOFMEASUREMENT-147) Created: 11/Jul/15  Updated: 12/Jul/15  Resolved: 12/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | class, conversion |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| A method call like  METRE.asType(Dimensionless.class);  should throw a ClassCastException but currently that does not happen. It worked in prior implementations of JSR-275 or JScience 5. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 12/Jul/15 ] |
| Still got this into 0.8-RC3 of RI. Key was, to expose Dimensionless constant ONE from the Units class, too like JScience5 did in its SI system class. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-146] [Create a single "internal" package and sub-packages](https://java.net/jira/browse/UNITSOFMEASUREMENT-146) Created: 10/Jul/15  Updated: 11/Jul/15  Resolved: 11/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC2](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17309) |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to JSR 354 RI "Moneta" or "Hibernate Validator" (RI for JSR 349: Bean Validation) it seems a good practice to structure internal implementation details under a single "internal" package. So changing e.g. tec.units.ri.format.internal to tec.units.ri.internal.format and similar. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-145] [Move Bootstrap mechanism into API](https://java.net/jira/browse/UNITSOFMEASUREMENT-145) Created: 10/Jul/15  Updated: 10/Aug/15  Resolved: 10/Aug/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17310) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-152~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-152) | Refactoring SPI | Resolved | | blocks | [~~UNITSOFMEASUREMENT-98~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-98) | Consider providing the "Units" class ... | Resolved | | blocks | [~~UNITSOFMEASUREMENT-151~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-151) | Define QuantityFactoryService in API | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-56~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-56) | Could we move AbstractQuantity into A... | Resolved | |  |
| **Tags:** | osgi, services, spi |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to the likes of JSR 354 (the actual class is so far heavily inspired by it) or others, e.g. 365 (CDI 2 and before) or 349 (Bean Validation 1.1) we shall offer a small simple Bootstrap mechanism in the actual JSR.  ServiceProvider already exists in the SPI, so the Bootstrap class should be moved from RI to API. Whether under that name or something better comes up, please have a look.  The approach by Bean Validation with a single class javax.validation.Validation looks most appealing, but looking at CDI, its main accessor class is javax.enterprise.inject.spi.CDI so it's equally possible to locate it in the SPI package. Since this package is optional in JSR 363 it leaves users of mandatory elements and maybe quantity types free to use other service and module systems like OSGi. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Jul/15 ] |
| JSR 107 offers one of the most basic ServiceLoader approaches with a single <https://github.com/jsr107/jsr107spec/blob/master/src/main/java/javax/cache/spi/CachingProvider.java> (though it has some form of optionality on the implementation side, but there is just 1 service, not 2 or more) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 26/Jul/15 ] |
| Using ServiceLoader means, the minimum Java version has to be 1.6 for API, since it wasn't introduced till Java 6. We already mention it in the Spec, and should someone really still use Java 5, they can use either JSR 275 or Unit-API "Legacy" 0.6 instead. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-144] [Failure to arithmetic operation](https://java.net/jira/browse/UNITSOFMEASUREMENT-144) Created: 07/Jul/15  Updated: 11/Jul/15  Resolved: 11/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107), [0.8-RC2](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17309) |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Critical |
| **Reporter:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This very basic code fails to do arithmetic add:  Quantity<Length> km = Quantities.getQuantity(1000, SI.METRE);  Quantity<Length> mile = Quantities.getQuantity(1, US.MILE);  System.out.println(km.add(mile));  this results in 1001 m, which is wrong. It should either convert automatically or throw some kind of exception of incompatible units. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Jul/15 ] |
| Most \*Quantity implementations in both RI and the SE port contain code like  Quantity<Q> converted = that.to(getUnit());  return Quantities.getQuantity(value + that.getValue().doubleValue(), getUnit());  in methods like add() but it should be something like  Quantity<Q> converted = that.to(getUnit());  return Quantities.getQuantity(value + converted.getValue().doubleValue(), getUnit()); |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Jul/15 ] |
| Should be mostly fixed in RI. As both "Full SI" and all non SI systems are no longer part of the RI (mainly for size reasons and to provide modularity in the spirit of "Jigsaw" or OSGi and other module systems) this can't be tested directly with the RI, thus <https://github.com/unitsofmeasurement/uom-systems/issues/6> was created to test the fix for this problem. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-143] [Multiples of HERTZ wrongly formatted](https://java.net/jira/browse/UNITSOFMEASUREMENT-143) Created: 03/Jul/15  Updated: 20/Oct/15  Resolved: 20/Oct/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC2](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17309) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Duplicate | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Cloners** | | | | | clones | [~~UNITSOFMEASUREMENT-140~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-140) | Unit HERTZ wrongly formatted | Resolved | | **Duplicate** | | | | | duplicates | [UNITSOFMEASUREMENT-166](https://java.net/jira/browse/UNITSOFMEASUREMENT-166) | Multiples of units need to be explici... | Open | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-78~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-78) | Can we improve Formatting by adding s... | Resolved | |  |
| **Tags:** | format |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At least 2 tests in <https://github.com/unitsofmeasurement/unit-ri/blob/master/src/test/java/tec/units/ri/format/UnitFormatTest.java> involving the unit HERTZ do not correctly show the symbol "Hz" or multiples like "MHz", but "1/s" and similar (e.g. "M1/s",...) This is mathematically correct, but especially the AlternateUnit applied in this case shall prefer the given symbol.  Verify for other AlternateUnit usages, if they are equally affected. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Oct/15 ] |
| The new ticket is more general and related to UnitFormat or implementing classes. |

|  |  |
| --- | --- |
| **[Define our criterion for inclusion / exclusion of quantity types](https://java.net/jira/browse/UNITSOFMEASUREMENT-100)** ([UNITSOFMEASUREMENT-100](https://java.net/jira/browse/UNITSOFMEASUREMENT-100)) [UNITSOFMEASUREMENT-142] [Document criterion for inclusion / exclusion of quantity types](https://java.net/jira/browse/UNITSOFMEASUREMENT-142) Created: 29/Jun/15  Updated: 20/Sep/15  Resolved: 20/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, document, documentation |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Sub-task for documenting changes in Spec or other relevent documents |

|  |  |
| --- | --- |
| **[Define our criterion for inclusion / exclusion of quantity types](https://java.net/jira/browse/UNITSOFMEASUREMENT-100)** ([UNITSOFMEASUREMENT-100](https://java.net/jira/browse/UNITSOFMEASUREMENT-100)) [UNITSOFMEASUREMENT-141] [Apply criterion for inclusion / exclusion of quantity types](https://java.net/jira/browse/UNITSOFMEASUREMENT-141) Created: 29/Jun/15  Updated: 01/Jul/15  Resolved: 01/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Sub-task to implement changes based on parent. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-140] [Unit HERTZ wrongly formatted](https://java.net/jira/browse/UNITSOFMEASUREMENT-140) Created: 25/Jun/15  Updated: 04/Jul/15  Resolved: 04/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC2](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17309) |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Cloners** | | | | | is cloned by | [~~UNITSOFMEASUREMENT-143~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-143) | Multiples of HERTZ wrongly formatted | Resolved | |  |
| **Tags:** | format |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At least 2 tests in <https://github.com/unitsofmeasurement/unit-ri/blob/master/src/test/java/tec/units/ri/format/UnitFormatTest.java> involving the unit HERTZ do not correctly show the symbol "Hz" or multiples like "MHz", but "1/s" and similar (e.g. "M1/s",...) This is mathematically correct, but especially the AlternateUnit applied in this case shall prefer the given symbol.  Verify for other AlternateUnit usages, if they are equally affected. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-139] [Special GRAM format problem with sub-multiples](https://java.net/jira/browse/UNITSOFMEASUREMENT-139) Created: 25/Jun/15  Updated: 10/Jul/15  Resolved: 10/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8-RC2](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17309) |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Duplicate** | | | | | duplicates | [~~UNITSOFMEASUREMENT-108~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-108) | getSymbol() and toString() differ for... | Closed | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-78~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-78) | Can we improve Formatting by adding s... | Resolved | |  |
| **Tags:** | format |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| SimpleUnitFormat and related UnitFormat implementations contain a special format handling of KILOGRAM vs. GRAM (since different unit systems consider each of them the Mass base unit) This does not work for most sub-multiples of GRAM, taking the following examples (also in various PrefixTest cases)  Quantity<Mass> m1 = Quantities.getQuantity(1.0, GRAM);  System.out.println(m1.toString());  Quantity<Mass> m2 = m1.to(MICRO(GRAM));  System.out.println(m2.toString());  gives  1.0 g  1000000.0 ng  The latter should be "µg". Similar results for NANO etc.  Also see <https://github.com/unitsofmeasurement/si-units/issues/4>  This only seems to affect Mass based on GRAM or KILOGRAM, other quantities/units are generally unaffected. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 06/Jul/15 ] |
| The best approach to this problem so far was to forward-engineer what JSR-275/JScience 4.3 did about it. They offered both UnitFormat.label() and UnitFormat.alias on an API level. Whether we require both is to be seen, but at least one way to tweak and improve formatting sounds like a good idea. |

|  |  |
| --- | --- |
| **[Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)** ([UNITSOFMEASUREMENT-3](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)) [UNITSOFMEASUREMENT-138] [Correct java.util.logging.Logger usages](https://java.net/jira/browse/UNITSOFMEASUREMENT-138) Created: 24/Jun/15  Updated: 30/Oct/15  Resolved: 30/Oct/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) | **Assignee:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| java.util.logging.Logger in Java ME has no "shortcut methods" like warning() and info() |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 25/Jun/15 ] |
| I guess we can close this, as it was already merged in master? |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-137] [dzone refcardz](https://java.net/jira/browse/UNITSOFMEASUREMENT-137) Created: 22/Jun/15  Updated: 02/Oct/16 | |
| **Status:** | Open |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | New Feature | **Priority:** | Trivial |
| **Reporter:** | [Rajmahendra Hegde](https://java.net/jira/secure/ViewProfile.jspa?name=rrhegde) | **Assignee:** | Unassigned |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | documentation | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |
| **Environment:** | Documentation | | |

|  |  |
| --- | --- |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| dzone refcardz  May be it is too early now but just for a backlog item.  Its good to have a refcardz once the JSR goes live. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-136] [Porting Spec on GitBook.com](https://java.net/jira/browse/UNITSOFMEASUREMENT-136) Created: 22/Jun/15  Updated: 17/Mar/16  Resolved: 17/Mar/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277), [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Trivial |
| **Reporter:** | [Rajmahendra Hegde](https://java.net/jira/secure/ViewProfile.jspa?name=rrhegde) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | documentation | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |
| **Environment:** | Documentation | | |

|  |  |
| --- | --- |
| **Tags:** | question, vote |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| I am sending there is big demand for collaborated documentation using GitBook.com  It provides markdown and asciidoc, download options with export pdf, epub, mobi or on-line view.  Some sample docs are :  <https://www.gitbook.com/book/gitbookio/javascript/details> <https://www.gitbook.com/book/gitbookio/documentation/details> |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Mar/16 ] |
| I think we should defer that to a future version of the JSR. Starting a new version (likely under a new JSR) we could look at either GitBooks or Asciidoc, etc. At the moment, all EC members who actively participate by writing or commenting are comfortable with Google Docs, so I would not change that now. |

|  |  |
| --- | --- |
| **[Website](https://java.net/jira/browse/UNITSOFMEASUREMENT-36)** ([UNITSOFMEASUREMENT-36](https://java.net/jira/browse/UNITSOFMEASUREMENT-36)) [UNITSOFMEASUREMENT-135] [Migrate from unitsofmeasurement.org](https://java.net/jira/browse/UNITSOFMEASUREMENT-135) Created: 11/Jun/15  Updated: 09/Nov/15  Resolved: 09/Nov/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | Maven, website |
| **Sprint:** | August, Q1/15, Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As Jean-Marie is likely to stop registering the legacy domain "unitsofmeasurement.org" content now deployed there needs to be relocated.  GitHub has a set of plugins for Maven site or other forms of deployment: <https://github.com/github/maven-plugins>  The content in question had to be published to <https://github.com/unitsofmeasurement/unitsofmeasurement.github.io> similar to e.g. <https://github.com/unitsofmeasurement/unitsofmeasurement.github.io/tree/master/pages> resulting in <http://unitsofmeasurement.github.io/pages/about.html> and similar pages.  JavaDocs should ideally be generated into a similar folder resulting in <http://unitsofmeasurement.github.io/apidocs> (this does not exist now) as Maven build jobs related to JavaDoc now assume this to hold JavaDocs.  Similar for Maven site if we also want to generate it there. It could be easier to try separate artifacts, so where under the old domain "jsr-363" is now generated, we could have a folder like  <http://unitsofmeasurement.github.io/unit-api/apidocs> (not existing) and others. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Sep/15 ] |
| It's <http://unitsofmeasurement.github.io/unit-api/site/apidocs/index.html> now.  Maven site is tweaked to use Maven GitHub plugins. To deploy the site succesfully, one must add settings.xml with working GitHub credentials according to: <https://github.github.com/maven-plugins/site-plugin/authentication.html>  Both worked for me locally. Either Martin should do the same for the GeoAPI Jenkins build allowing updates of the GitHub pages. As an alternative, we could try building it on Circle-CI, similar to how all Snapshots are already published to Bintray (using a similar API Key/token) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 21/Sep/15 ] |
| It seems, like a single build command mvn clean install deploy site site:deploy --show-version --batch-mode as <http://jenkins.geotoolkit.org/job/JSR-363/177/console> suggests is too big. At least the new "site" replacement doesn't understand "deploy" at all, nor does it look relevant (AFAIK this pushes the artifacts to the snapshot repo) Ideally "install" and other steps like "site" or "deploy" should be chained, similar to e.g. Circle-CI does for the RI, see <https://circleci.com/gh/unitsofmeasurement/unit-ri/17> Where necessary build steps or plugins that aren't compatible with each other should be separated into a different Maven profile. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 05/Oct/15 ] |
| Now that JSR-363 has its own Continuous Integration server, the JSR-363 job on [http://jenkins.geotoolkit.org](http://jenkins.geotoolkit.org/) and the [http://www.unitsofmeasurement.org](http://www.unitsofmeasurement.org/) web pages have been deleted. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 05/Oct/15 ] |
| At the moment the final site-deploy step is hindered by GitHub's insufficient "abuse detection" or some kind of quota, but we can live with API JavaDoc uploaded manually when important structural changes took place. Everything else, especially snapshot deployment is automated. As soon as a new Maven Site with updated CI server info can be pushed to GitHub, I guess we can close this. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 09/Nov/15 ] |
| While the site generation is currently manual (due to Github flaw in "abuse detection" everything else worked properly and is now using the new github.io pages. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-134] [Updating pom file(s) to use latest felix plugin](https://java.net/jira/browse/UNITSOFMEASUREMENT-134) Created: 29/May/15  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200), [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | Maven, osgi |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As seen in <https://issues.jboss.org/browse/CDI-509> for CDI 2 EDR 1, there's a new Apache Felix plugin version. We should leverage that, and while version numbers of individual packages are probably not so different in our case, also adjust OSGi export-package similar to:  <Export-Package>  javax.decorator;version=1.1,  javax.enterprise.\*;version=1.1,  javax.decorator;version=2.0,  [...]  </Export-Package> |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 28/Jun/15 ] |
| Actually we are using Felix 2.5.3 plugin already, so further adjustments can be done for PDR |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-133] [GeoAPI Maven Snapshot repo is not updated when Site is](https://java.net/jira/browse/UNITSOFMEASUREMENT-133) Created: 19/Apr/15  Updated: 29/May/15  Resolved: 19/Apr/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | maven | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | maven, repository |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| When Jenkins on GeoAPI builds the API, the site (<http://www.unitsofmeasurement.org/jsr-363/index.html>) gets updated, but since March 23 it seems to have stopped deploying to Maven. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 19/Apr/15 ] |
| Deployment was temporarily disabled when the JSR-363 version was set to 0.8-RC1 because the Jekins server can not deploy on non-snapshot repositories. But since JSR-363 switched back on 0.8-SNAPSHOT on April 14, we can re-enable the deployment. I made the change and verified that the JAR files has been deployed. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/May/15 ] |
| The only time when API will be either 0.8 or 0.8-RCx is right before a version is tagged, but we can ensure, the version is always moved back to -SNAPSHOT until the JSR is final. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-132] [Could we rename SIPrefix to MetricPrefix?](https://java.net/jira/browse/UNITSOFMEASUREMENT-132) Created: 13/Apr/15  Updated: 26/Aug/15  Resolved: 09/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-126~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-126) | Should we rename SI to Metric? | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-157~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-157) | Shall we keep the name MetricPrefix? | Resolved | |  |
| **Tags:** | design, naming |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As per Wikipedia <http://en.wikipedia.org/wiki/Metric_prefix> the page is called MetricPrefix, so where prefixes remain in the RI (bound to ME-compatible arithmetics it would make sense) using this term does not contradict the SI system, but keeps the name independent of it. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-131] [Current DimensionalModel gets cleaned up erroneously](https://java.net/jira/browse/UNITSOFMEASUREMENT-131) Created: 09/Apr/15  Updated: 08/Jun/15  Resolved: 08/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [sebster](https://java.net/jira/secure/ViewProfile.jspa?name=sebster) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The Current static (should it be CURRENT to comply with standard Java conventions?) DimensionalModel in the DimensionalModel class is held by a WeakReference. When there are no more references to the value held by the WeakReference, it is garbage collected and the DimensionalModel.current() method will return null subsequently. This will cause methods such as AbstractUnit.getConverterToAny(...) to fail with a NullPointerException. Depending on the JDK and memory availability, the weak reference is garbage collected more or less aggressively (on my Oracle JDK6 it worked fine, but the IBM JDK failed consistently).  The patch I currently used is as follows (I don't really understand the need for a WeakReference here):  @ -25,8 +25,6 @@ \*/ package tec.units.ri;  -import java.lang.ref.Reference; -import java.lang.ref.WeakReference; import java.util.Map;  import javax.measure.Dimension; @@ -81,7 +79,7 @@ public abstract class DimensionalModel { /\*\*   * Holds the current model. \*/ * private static Reference<DimensionalModel> Current = new WeakReference<DimensionalModel>(new StandardModel()); + private static DimensionalModel CURRENT = new StandardModel();   /\*\*   * Returns the current dimensional model @@ -90,7 +88,7 @@ public abstract class DimensionalModel { * @return the current physical model. \*/ public static DimensionalModel current() { - return DimensionalModel.Current.get(); + return DimensionalModel.CURRENT; }   // /\*\* |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Apr/15 ] |
| Will check it out. If the patch works within the boundaries of Java ME 8, it should work. As for current() if the worry was why it's not getCurrent() this was simply modeled after a central CDI class: <http://docs.jboss.org/cdi/api/1.1/javax/enterprise/inject/spi/CDI.html> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Jun/15 ] |
| The reference was closest to JScience contexts being used there, but simply returning a constant value like your patch suggested works, so it should solve the problem. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-130] [Infinite recursion in AbstractConverter](https://java.net/jira/browse/UNITSOFMEASUREMENT-130) Created: 09/Apr/15  Updated: 09/Jun/15  Resolved: 09/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [sebster](https://java.net/jira/secure/ViewProfile.jspa?name=sebster) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 1 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Problem: convert(Number) calls itself.  Patch:  @@ -100,7 +100,7 @@ public abstract class AbstractConverter implements UnitConverter {  @Override public Number convert(Number value)  { - return convert(Double.valueOf(value.doubleValue())); + return convert(value.doubleValue()); }  public abstract double convert(double value); |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-129] [JDK6 compatibility for unit-ri](https://java.net/jira/browse/UNITSOFMEASUREMENT-129) Created: 09/Apr/15  Updated: 02/Dec/15  Resolved: 02/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16915) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [sebster](https://java.net/jira/secure/ViewProfile.jspa?name=sebster) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Attachments:** | java-se6.patch |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-169~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-169) | Consider additional profile(s) for TCK | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-125~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-125) | Some API JUnit tests won't work under... | Resolved | |  |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| I've made RI compatible with Java SE 6 (mvn clean install runs against 1.6.0u45).  I've made the minimum number of possible changes to keep the patch as clean as possible.  I've chosen to "inline" the references to the Objects utility class instead of making an own Objects utility class with only the needed methods. The only real impact is the readability of Objects.equals but I believe the readability is not too bad as is. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 09/Apr/15 ] |
| Great, do you have a patch/diff to see changes? |
| Comment by [sebster](https://java.net/jira/secure/ViewProfile.jspa?name=sebster) [ 09/Apr/15 ] |
| For lack of a better method at the moment, patch at: <https://www.sebster.com/~sebster/java-se6.patch> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 09/Apr/15 ] |
| Thanks. Will check it out, and if you don't mind, I'd merge it over to this JIRA ticket, too (strange even the reporter of a ticket can't attach anything, will explore with Java.net crew why) |
| Comment by [sebster](https://java.net/jira/secure/ViewProfile.jspa?name=sebster) [ 09/Apr/15 ] |
| Yeah, no problem. I'm happy if I can contribute. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Apr/15 ] |
| Patch provided by reporter. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/Jul/15 ] |
| I'm afraid, after sometimes massive refactoring the patch will no longer be of much use. I briefly tried to switch the JDK version to 6 and a main reason it does not work is the Objects class used in many places. Other J7 dependencies like "Diamond" seem to be gone pretty much all of them. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Nov/15 ] |
| Objects.equals() seems pretty much a one-liner, so substituting it looks very easy. Not even sure, why they took the trouble to add that to the JDK? |
| Comment by [sebster](https://java.net/jira/secure/ViewProfile.jspa?name=sebster) [ 23/Nov/15 ] |
| I have a new patch (done today) based on the 0.8 tag. I've provided it here: <http://www.sebster.com/~sebster/0001-Java-6-compatibility.patch>  We (you won't believe this) also need to port it to Java 5, and that's exactly the same as porting it to Java 6 except that the ServiceLoader class does not exist in Java 5 (used once). I still have to figure out how I can replace that. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Nov/15 ] |
| Thanks, will check it out. @otaviojava added a few "diamond" cases that need to be built back recently, but other than that it should be pretty Java 6 compliant. As for Java 5, I don't know if we'd support that out of the box, given the API also uses ServiceLoader present from Java SE 6 on and also in CLDC8.  It depends what if anything you'd need from the SPI: <http://unitsofmeasurement.github.io/unit-api/site/apidocs/javax/measure/spi/package-frame.html> Everything except SystemOfUnits (we want to give people a choice to use other forms of organizing units in a "DB" or collection form) probably won't make too much sense without ServiceLoader, so you could leave the optional SPI package out and still be compliant with your own implementation |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Nov/15 ] |
| Ok, I was able to undo all the Diamond stuff and apply the patch. The RI builds against Java 6 now by default, too.  The API got some profiles, on one hand to override the default JDK (for some plugins it's Java 7 for the API, but the CI build uses 6) allowing in theory to build with Java 5 to 8 compatibility. The problem is, I installed the latest Java 5 SDK (1.5\_u22) available and Eclipse could still handle it, but Maven has more than one plugin that simply won't work below Java 6 (for that I had to down-grade the Surefire plugin by default)  So while everything other than SPI/Service Loader functionality should be Java 5 compatible from all we know (JSR-275 and Unit-API 0.6 both still built Java 5 compatible) for build tools you're on your own. Not sure, if Gradle ever supported such an old Java version at all? Otherwise please try Ant, it will of course work,, but for those build scripts you're on your own when it comes to building the API (core or other JARs) with Java 5 for Java 5.  Same for a Java 5 compliant fork of the RI. Most of it should work, but we can't provide it out of the box any more I'm afraid. |
| Comment by [sebster](https://java.net/jira/secure/ViewProfile.jspa?name=sebster) [ 24/Nov/15 ] |
| Thanks for applying the patch. Do you use the maven toolchains plugin? Because from maven 3.3 onward I believe that maven even won't compile projects with jdk6 without it. With the maven toolchains plugin it's possible to compile projects even with jdk5 (that's what I'm doing now). However that wouldn't solve the ServiceProvider issue. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Nov/15 ] |
| No, but I know, Apache Tamaya where at least 2 EG members including myself also @otaviojava) are committers, does. If it adds a benefit and does not break anything, we might explore that for the Final Draft iteration/API.  Regarding ServiceProvider, if you don't need SystemOfUnits too badly (adding it to core would put a burden on those who e.g. want to declare a Unit as enum and don't need to implement it, that's why it's part of the SPI) you can stick to another profile like "base\_quantity" or "quantity" plus "format" (we may not have the combination yet since they are independent, but that's probably something to add, at least for all of format+quantity, base\_quantity requires to pick some quantities anyway, it's meant for very small devices as an option) and not use the "spi" package. Does that help? (you could simply fork or clone the SystemOfUnits interface in your own app if you need it, we'd rather not force everyone to use it if they only need Unit and Quantity or chose their own approach to grouping/storing units)  The maven-jar-plugin with qualifiers like "core" is a compromise. In an early version of the API I actually grouped them into real Maven modules for "core", "format",..., then putting the whole API in a single JAR via maven-shade-plugin but some EG members felt a bit uneasy about it, so we changed the structure. Mostly to test the TCK and smaller implementations (e.g. <https://github.com/unitsofmeasurement/uom-impl-enum> currently implements "format" + "base\_quantity", we may extend it to a new "format+quantity" profile) module JARs are also good to have, but as of now, we probably won't deploy them to Maven repositories. I think the maven-jar-plugin (or rather maven-compile-plugin) compiles everything and then only puts selected packages to a smaller archive, thus compiling against Java 5 may be tricky, but as mentioned in the JSR 354 list, could you try compiling with Java 6 using the "jdk5" option? The JDK option, not the JDK running it should make the JARs compatible with a minimum Java version. As long as you avoid using the "spi" JAR under Java 5, the others should work. The problem with Java 5, unlike 6-8, it's fully end of life, see: <http://www.oracle.com/technetwork/java/eol-135779.html> Especially in the Embedded world taking at least "Extended Support" into consideration is more than fair. And Java 6 still falls into that till December 2018, 2 years after what we estimate to be our Final schedule (Q3 or Q4/16 sound reasonable, depending on how PR goes and what feedback we get) while this support for Java 5 has ended. If you want to suggest e.g. the Maven Toolchain, please feel free to do so (at least for API first), we have other proposals in the backlog, but some hints for tools or new formats (like Asciidoc, etc.) we cannot promise to cover all of them, e.g. if they only meet a very small niche. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Nov/15 ] |
| P.s.: Do you have a GitHub user, too? Would be happy to add you as "Observer" or "Adopter" (that sounds more appropriate, also gives commit rights to most repos except the core JSR artifacts like API, RI or TCK which require EG membership) |
| Comment by [sebster](https://java.net/jira/secure/ViewProfile.jspa?name=sebster) [ 24/Nov/15 ] |
| Yes, my github user is sebster as well. I think it's already added to many of the projects.  Regarding the toolchains plugin: it adds a benefit that it actually compiles with the JDK of the required source level, but it also requires you to specify the jdk locations in a settings.xml, so it makes it harder for someone to build the software. I don't know what maven version you are using, but currently, maven 3.3+ does not support building with JDK6 anymore, so to officially support JDK6 I think the toolchains plugin is required. This is also currently a problem with the maven artifacts in the maven repositories for unit-api and unit-ri: they are compiled with JDK7 even if they have source level JDK6, and as such, when trying to use them in a JDK6 project, you get a class version error: Unsupported major.minor version 51.0. (Note: unit-api compiles in JDK6 without any source changes, but the current repo artifact is compiled with JDK7 and therefore not usable with JDK6). I have attached the diffs for the toolchains plugin though I am having a bit of a fight with the javadoc plugin because it's calling the proper javadoc but passing it a JDK8 option (-Xdoclint:none) since maven is running under JDK8 and thinks that javadoc supports it; I haven't fixed that problem yet. (<http://www.sebster.com/~sebster/0001-Make-api-java-6-compatible-and-use-toolchains-plugin.patch>)  As for JDK5, I got it working by basically removing the non-compiling stuff from the spi package. I understand Java 5 is fully unsupported and end of life, and I agree with you that it's probably not a good idea for you guys to go to the effort of supporting it. I'm going to maintain my own fork for this, mainly it's just throwing some stuff out, so it should be easy for me to pull in upstream changes for new releases. I'm unhappy that I need to support Java 5 and hopefully it's not for more than a year or two, but for now that's life. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Nov/15 ] |
| Will check, somehow it wasn't found, but maybe you are already in a team (Adopters, so you can contribute to all the mentioned projects ) As for 0.8 (PR) especially the RI was compiled with Java 7 compatibility, and it's possible I also did so for the API (which I have to sign on the local machine before deploying it to JCenter) Have a look at <https://oss.jfrog.org/oss-snapshot-local/tec/units/unit-ri/0.9-SNAPSHOT/> and <https://oss.jfrog.org/artifactory/oss-snapshot-local/javax/measure/unit-api/0.9-SNAPSHOT/>, they are all built against Java 6 only. Please try what you did before.  If the JDK location for the toolchain plugin must be set in the settings.xlm that sounds very hard to comply with the DevOps/Docker style CI tools like Circle-CI we use everywhere (some repos have others as fall-back, but all of them build on Circle now at UoM) In theory something like that could be available via CIRCLE\_... variables, but in most cases that's up to the container and build job, all you want to chose is the JVM version. If you can handle Java 5 the way you did (chosing the right profile, I'd say you may even be quite likely to pass the TCK for particular profiles other than SPI) that sounds good. If there is no clash with other environments, e.g. Java ME Embedded (there it currently works best to have API, RI and your solution all from the source, see <https://github.com/unitsofmeasurement/uom-demos/tree/master/javame/medemo>) we should be able to keep both API and RI binary compatible with Java SE 6. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Dec/15 ] |
| RI now runs properly against Java SE 6 |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-128] [Add ServiceProvider interface](https://java.net/jira/browse/UNITSOFMEASUREMENT-128) Created: 09/Apr/15  Updated: 09/Jun/15  Resolved: 09/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | New Feature | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-99~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-99) | Rename javax.measure.spi.\*Service as ... | Resolved | |  |
| **Tags:** | api, design, spi |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to JSR 354 (using the BP <https://github.com/JavaMoney/jsr354-api-bp/blob/master/src/main/java/javax/money/spi/ServiceProvider.java> as it's Java 6/7 compliant) it seems good to have a ServiceProvider interface. That may be the only "Provider" for the SPI, sticking to the "\*Service" naming convention for other components, since ServiceProvider then literally provides services rather than again providers Implementing this unlike 354 might be up to RI or other implementations. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 09/Apr/15 ] |
| Changing all names from \*Service to \*Provider wouldn't make sense if ServiceProvider in a way 354 applies it was introduced. Since it provides those \*Service elements |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-127] [Simplify unit systems in RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-127) Created: 08/Apr/15  Updated: 01/Jul/15  Resolved: 01/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-121~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) | Introduce "Pluggable" Unit Systems an... | Resolved | | **Duplicate** | | | | | is duplicated by | [~~UNITSOFMEASUREMENT-126~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-126) | Should we rename SI to Metric? | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-100~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) | Define our criterion for inclusion / ... | Resolved | |  |
| **Tags:** | design, unit |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| To clarify the SI-compliant, but "simplified" scope of API quantities, and align naming also a bit with the Quantities accessor class, I suggest we simplify and combine unit systems in the RI into a single Units class. (which implements SystemOfUnits or whatever the final name for it) ICU4J or JSR 256 have static accessors to known units in the actual type like <http://icu-project.org/apiref/icu4j/com/ibm/icu/util/MeasureUnit.html> We can't do that in Unit and it would be dirty, but a collecting class like Units sounds appropriate. Ideally merging the "unit" and "spi" package. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Apr/15 ] |
| This is more a follow-up than duplicate, but related to changing the names of unit systems in the RI. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Apr/15 ] |
| Effectively renaming CommonUnits to Units and merging other systems into it. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-126] [Should we rename SI to Metric?](https://java.net/jira/browse/UNITSOFMEASUREMENT-126) Created: 02/Apr/15  Updated: 15/Apr/15  Resolved: 07/Apr/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Duplicate** | | | | | duplicates | [~~UNITSOFMEASUREMENT-127~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-127) | Simplify unit systems in RI | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-132~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-132) | Could we rename SIPrefix to MetricPre... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-103~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) | Look into Unicode CLDR | Resolved | |  |
| **Tags:** | design, naming |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At least JScience 5 called the SI unit system Metric since end of JSR 275. ICU4J on one hand has at least one internal class calling it "SI", too (a reason why e.g. UOMo stuck to that name, but the underlying Unicode CLDR calls the system "Metric" again, see: <https://github.com/unicode-cldr/cldr-units-modern/blob/master/main/en/measurementSystemNames.json>  A side-effect could be improved browsability by IDEs like Eclipse. A class name that's only 2 or 3 characters long is assumed a camelcase short-cut, and it usually gets confused when trying to browse a type "SI". If we made any change, I'd stick exactly to the 3 standard names in CLDR.  WDYT? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Apr/15 ] |
| Actually Wikipedia under "SI Units" <http://en.wikipedia.org/wiki/International_System_of_Units> claims **SI** is the **modern form** of the <http://en.wikipedia.org/wiki/Metric_system> so "Metric System" is a whole variety of standardization efforts, while "SI" marks its latest stage. So let's keep SI unless you really want to change it? |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-125] [Some API JUnit tests won't work under Java 6](https://java.net/jira/browse/UNITSOFMEASUREMENT-125) Created: 22/Mar/15  Updated: 20/Sep/15  Resolved: 20/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-129~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-129) | JDK6 compatibility for unit-ri | Resolved | |  |
| **Tags:** | junit, test\_issue |
| **Epic Link:** | [Test](https://java.net/jira/browse/UNITSOFMEASUREMENT-111) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This came up when running the API on drone.io against Java 6 (officially the minimum version) see <https://drone.io/github.com/unitsofmeasurement/unit-api/9>  At least some JUnit tests don't seem to run under SE 6 as they do on 7 or above. Not the highest priority, especially since SE 6 is no longer officially supported by Oracle. Nor looking like a problem of the actual API deliverables, but if someone has time and wants to look into it, feel free.  $ mvn install -q -DskipTests=true  [ERROR] COMPILATION ERROR :  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/DistanceQuantity.java:[162,38] inconvertible types  found : javax.measure.test.quantity.DistanceQuantity  required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/AreaQuantity.java:[151,38] inconvertible types  found : javax.measure.test.quantity.AreaQuantity  required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/VolumeQuantity.java:[150,38] inconvertible types  found : javax.measure.test.quantity.VolumeQuantity  required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/TimeQuantity.java:[136,38] inconvertible types  found : javax.measure.test.quantity.TimeQuantity  required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/BitQuantity.java:[159,38] inconvertible types  found : javax.measure.test.quantity.BitQuantity  required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/BitRateQuantity.java:[145,38] inconvertible types  found : javax.measure.test.quantity.BitRateQuantity  required: javax.measure.Quantity<T>  [ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:testCompile (default-testCompile) on project unit-api: Compilation failure: Compilation failure:  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/DistanceQuantity.java:[162,38] inconvertible types  [ERROR] found : javax.measure.test.quantity.DistanceQuantity  [ERROR] required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/AreaQuantity.java:[151,38] inconvertible types  [ERROR] found : javax.measure.test.quantity.AreaQuantity  [ERROR] required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/VolumeQuantity.java:[150,38] inconvertible types  [ERROR] found : javax.measure.test.quantity.VolumeQuantity  [ERROR] required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/TimeQuantity.java:[136,38] inconvertible types  [ERROR] found : javax.measure.test.quantity.TimeQuantity  [ERROR] required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/BitQuantity.java:[159,38] inconvertible types  [ERROR] found : javax.measure.test.quantity.BitQuantity  [ERROR] required: javax.measure.Quantity<T>  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/unit-api/src/test/java/javax/measure/test/quantity/BitRateQuantity.java:[145,38] inconvertible types  [ERROR] found : javax.measure.test.quantity.BitRateQuantity  [ERROR] required: javax.measure.Quantity<T>  [ERROR] -> [Help 1] |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Sep/15 ] |
| Circle-CI <https://circleci.com/gh/unitsofmeasurement/unit-api> now runs with Java 6 |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-124] [Introduce a Unit "DB"](https://java.net/jira/browse/UNITSOFMEASUREMENT-124) Created: 04/Mar/15  Updated: 30/Dec/15  Resolved: 30/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-89~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-89) | SystemOfUnits.getUnits() needs to def... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-88~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-88) | SystemOfUnits need to tell its base u... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-78~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-78) | Can we improve Formatting by adding s... | Resolved | | relates to | [~~UNITSOFMEASUREMENT-161~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-161) | Do we also want UnitFormat.alias()? | Resolved | |  |
| **Tags:** | design, question, vote |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to UCAR Units (derived from JSR 108 lead by UCAR) it may help to introduce a sort of unit "database" like its UnitSystem <https://www.unidata.ucar.edu/software/thredds/v4.3/netcdf-java/v4.0/javadocAll/ucar/units/UnitSystem.html> returns.  UnitDB (the name is questionable, some suggestions below) provides access to certain groups of units, UCAR has a choice of **Base Units** and **All Units** of a system, that also goes along issues (links -88 and -89) raised before.  The name UnitDB suggests it is a (relational) database, which often is not the case. Some possible names are   * UnitData * UnitGroup * UnitInfo * UnitRecord   Feel free to suggest a better name you may think of. Please vote if you care with a +1 and (optionally) the name you think is best. If you prefer a different approach and don't think, a "DB" layer is necessary below a unit system, please vote -1.  Thanks, Werner |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Mar/15 ] |
| Methods like label() or alias() in UnitFormat meet a similar goal, but at most those would be specific to a particular format or (where formats are created by Locale) locale. Otherwise apply to the whole instance of the VM, not a particular unit system or source. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 30/Dec/15 ] |
| May pick this idea up later, but for 1.0 it is out of scope. |

|  |  |
| --- | --- |
| **[Make RI and related libraries available to public Maven repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-35)** ([UNITSOFMEASUREMENT-35](https://java.net/jira/browse/UNITSOFMEASUREMENT-35)) [UNITSOFMEASUREMENT-123] [Make libraries available on MavenCentral](https://java.net/jira/browse/UNITSOFMEASUREMENT-123) Created: 04/Mar/15  Updated: 25/Jun/15  Resolved: 25/Jun/15 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Lib](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15569) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Cloners** | | | | | clones | [~~UNITSOFMEASUREMENT-84~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-84) | Make RI available on MavenCentral | Resolved | |  |
| **Tags:** | deployment, maven |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Jun/15 ] |
| uom-lib is already available on JCenter, closing this as it will be tracked further on GitHub, e.g. <https://github.com/unitsofmeasurement/uom-lib/issues/9> |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-122] [Should we rename SystemOfUnits?](https://java.net/jira/browse/UNITSOFMEASUREMENT-122) Created: 02/Mar/15  Updated: 09/Jun/15  Resolved: 26/Apr/15 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-88~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-88) | SystemOfUnits need to tell its base u... | Resolved | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-91~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-91) | Add isSystemAvailable(), getAvailable... | Resolved | | blocks | [~~UNITSOFMEASUREMENT-121~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) | Introduce "Pluggable" Unit Systems an... | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-99~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-99) | Rename javax.measure.spi.\*Service as ... | Resolved | |  |
| **Tags:** | design, naming, question, vote |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Going over an abstraction similar to DataSource > Connection if you want in SQL terms, allowing easier and more dynamic unit definitions, the question also comes up, if we want to keep the name SystemOfUnits or consider alternatives like   * UnitSystem (similar to UCAR, derived from JSR 108) * MeasurementSystem (used by ICU4J)   For providing services like (currently named) SystemOfUnitsService as opposed to UnitFormatService names like UnitSystem and UnitSystemService also sound more harmonically.  Please advise below with a simple +1 for an option you prefer. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [anakarparida](https://java.net/jira/secure/ViewProfile.jspa?name=anakarparida) [ 02/Mar/15 ] |
| +1 UnitSystemService |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Mar/15 ] |
| I guess you mean UnitSystem? (the \*Service would come naturally for related classes)  Sorry if the list of classes below may be irritating, but I guess we get the choice you prefer. |
| Comment by [anakarparida](https://java.net/jira/secure/ViewProfile.jspa?name=anakarparida) [ 02/Mar/15 ] |
| Yes UnitSystem is the preferred choice. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Mar/15 ] |
| +1 SystemOfUnits (on behalf of dautelle) |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 20/Apr/15 ] |
| I like SystemOfUnits better, no scientific point; but UnitSystem sounds "more right".  It's hard to know if it's right because I'm no native english speaker, and System of Units translates directly to Portuguese so I tend to like it better. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Apr/15 ] |
| So you "abstain" or prefer a "0" vote?  I know it is tricky, Wikipedia actually calls it "System of Measurement" <http://en.wikipedia.org/wiki/System_of_measurement> See MeasurementSystem, by ICU4J. UCAR, the successor to work done at JSR-108 once now calls it UnitSystem. I guess the term SystemOfUnits came with JSR-275. And at least this scientific modelling library from Switzerland written in C (<http://geant4.web.cern.ch/geant4/>) has a C header file: <http://geant4.web.cern.ch/geant4/reports/gallery/electromagnetic/units/SystemOfUnits.html>. So far there are 2 +1 votes for each, I'll leave my vote to decide it if Leonardo abstains. Everyone should have an equal vote here, so please feel free to still chip in if you can |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 20/Apr/15 ] |
| I vote on SystemOfUnits, unless it's wrong Engrish |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 20/Apr/15 ] |
| Well, while browsing for the SI x Metric, I found that SI is [International System of Units](http://en.wikipedia.org/wiki/International_System_of_Units). So I stick to my vote! |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Apr/15 ] |
| No it's not. Thanks for your vote. I'll simply **abstain/0** to respect the votes by 2 other Spec Leads (against one other vote for "UnitSystem")  We have somewhat big names in SPI like SystemOfUnitsService but tere are separate discussions to chang the \*Service postfix to something else. So let's discuss it there based on SystemOfUnits. Will leave it open till the end of the day and then close if there are no further significant changes.  Thanks, Werner |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 26/Apr/15 ] |
| Vote concluded to keep it SystemOfUnits |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-121] [Introduce "Pluggable" Unit Systems and data sources](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) Created: 20/Feb/15  Updated: 23/Feb/16  Resolved: 23/Feb/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [Lib](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15569), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Attachments:** | dependencies.png |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-127~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-127) | Simplify unit systems in RI | Resolved | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-122~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-122) | Should we rename SystemOfUnits? | Closed | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-152~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-152) | Refactoring SPI | Resolved | | is related to | [~~UNITSOFMEASUREMENT-120~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-120) | Sync Unit Systems with oBIX | Resolved | | is related to | [~~UNITSOFMEASUREMENT-103~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) | Look into Unicode CLDR | Resolved | | is related to | [~~UNITSOFMEASUREMENT-98~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-98) | Consider providing the "Units" class ... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-100~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) | Define our criterion for inclusion / ... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-56~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-56) | Could we move AbstractQuantity into A... | Resolved | |  |
| **Tags:** | data, i18n/l10n, spi |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| To overcome problems with different unit systems or data sources for unit definitions (e.g. UCUM vs. SI, etc.) we should try to make this somewhat pluggable, see **i18n** enhancements by Java 8 using CLDR among other sources: <http://docs.oracle.com/javase/8/docs/technotes/guides/intl/enhancements.8.html>  Allowing something like:  "javax.measure.unitsystemproviders=NIST,SPI,SI,CLDR,UCUM"  The default behavior is equivalent to the following setting:  "javax.measure.unitsystemproviders=NIST,SPI"  Not sure, if we'd call the default "NIST" or something else (like "DEFAULT") that's up to working on this item. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Mar/15 ] |
| Not a real show-stopper, the type works with either name, but if we feel there's a better name, it can be changed more easily prior to Public Draft. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Jun/15 ] |
| This is an overview of possible dependencies between API, implementations, libraries (especially Unit Systems or similar) and applications consuming them |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 23/Feb/16 ] |
| Not sure how to make this in Java ME... |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Feb/16 ] |
| Unlike JSR 354 where Conversion providers were included with the very core of the RI (similar to Date/Time in Java 8 or ever since) we pretty much decoupled this into "extension modules" so I think other than maybe Bootstrap helping to retrieve certain implementations (e.g. UCUMFormat in the UCUM module (currently for SE only) we don't seem to have such a strong requirement here any more. If the Liblets concept works similar to Java SE service loaders, it should work under ME, but the core API and SPI are rather small now. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Feb/16 ] |
| Most of this was deferred into extension modules like <https://github.com/unitsofmeasurement/si-units> or <https://github.com/unitsofmeasurement/uom-systems>  Mostly via the Bootstrap class and services it provides, codes like "UCUM" or similar may be used, and methods like getAvailableFormatNames() in UnitFormatService provide a list of services that are available/loaded. It does not seem necessary to define any other loading mechanism than what's used by an implementation, either Java Service Loaders, (ME) LibLets, OSGi or in future maybe Jigsaw. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-120] [Sync Unit Systems with oBIX](https://java.net/jira/browse/UNITSOFMEASUREMENT-120) Created: 16/Feb/15  Updated: 07/Apr/15  Resolved: 07/Apr/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-103~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) | Look into Unicode CLDR | Resolved | | is related to | [~~UNITSOFMEASUREMENT-116~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-116) | Refactor UCUMFormat into a locale neu... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-121~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) | Introduce "Pluggable" Unit Systems an... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-100~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) | Define our criterion for inclusion / ... | Resolved | |  |
| **Tags:** | resources, unit |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Especially for Smart Home and Smart Buildings the OASIS standard oBIX (<http://www.obix.org/>) is quite important for communications between vendors and devices. oBIX has a very well-defined set of units based on at least 2 elements Unit and Dimension that pretty much reassemble those defined by JSR 363. Project Haystack (<http://project-haystack.org/doc/Units>) contains a simplified plain text format of these units based on the Fantom language (its co-creator Brian Frank was in the JSR 275 EG through Tridium, Inc, so the way Fantom supports these units is very much inspired and derived on JSR 275, too ) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Feb/15 ] |
| Both are possible sources to tap into. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Mar/15 ] |
| oBIX where fully supported may change the number of quantities. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Apr/15 ] |
| At least for the RI oBIX contains too many non-SI quantities (see files attached to <https://java.net/jira/browse/UNITSOFMEASUREMENT-100>), so we won't support it here. Subject to UOM-SE, or Tools: <https://github.com/unitsofmeasurement/uom-tools/issues> |

|  |  |
| --- | --- |
| **[Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)** ([UNITSOFMEASUREMENT-3](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)) [UNITSOFMEASUREMENT-119] [Fix L10nPropertyResources](https://java.net/jira/browse/UNITSOFMEASUREMENT-119) Created: 08/Feb/15  Updated: 11/Jun/15  Resolved: 11/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | embedded, i18n/l10n, l10n, portability |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| To get independent of SE ResourceBundle/Locale a small interface L10nBundle is implemented by 2 classes. One works, MapResourceBundle is inspired by Nokia suggestions (see attachments/comments of [UNITSOFMEASUREMENT-86](https://java.net/jira/browse/UNITSOFMEASUREMENT-86)), it uses the idea of holding resource data in classes like the JDK class resource bundle. L10nResources largely based on an ME substitution for bundles backed by Properties files doesn't work.  If there's no performance or size penalty, using properties would be nice, otherwise we may have to select the one that works. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/Jun/15 ] |
| As of now the class-based L10nResources work, so L10nPropertyResources is not really neccessary. |

|  |  |
| --- | --- |
| **[Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)** ([UNITSOFMEASUREMENT-101](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)) [UNITSOFMEASUREMENT-118] [Add Tests according to Spec Chapter 4](https://java.net/jira/browse/UNITSOFMEASUREMENT-118) Created: 27/Jan/15  Updated: 03/Apr/16  Resolved: 03/Apr/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-159~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-159) | Move test-audit.xml to main/resources | Resolved | |  |
| **Tags:** | api, spec, tck-red, test, testing |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Spec Chapter 4 (API) requires TCK Tests   * Assess if and which tests are required * Where necessary update test-audit.xml of the TCK project * Write tests for sections identified as relevant |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Apr/16 ] |
| This looks pretty complete |

|  |  |
| --- | --- |
| **[Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)** ([UNITSOFMEASUREMENT-101](https://java.net/jira/browse/UNITSOFMEASUREMENT-101)) [UNITSOFMEASUREMENT-117] [Add Tests according to Spec Chapter 2](https://java.net/jira/browse/UNITSOFMEASUREMENT-117) Created: 27/Jan/15  Updated: 20/Sep/15  Resolved: 20/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | tck-red, test, testing |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Spec Chapter 2 (Definitions) could require TCK Tests   * Assess if and which tests are required * Where necessary update test-audit.xml of the TCK project * Write tests for sections identified as relevant |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Sep/15 ] |
| As of now only Chapter 4 and 5 seem TCK relevant |

|  |  |
| --- | --- |
| **[Remove UCUM from RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-114)** ([UNITSOFMEASUREMENT-114](https://java.net/jira/browse/UNITSOFMEASUREMENT-114)) [UNITSOFMEASUREMENT-116] [Refactor UCUMFormat into a locale neutral SimpleUnitFormat](https://java.net/jira/browse/UNITSOFMEASUREMENT-116) Created: 22/Jan/15  Updated: 11/Jul/15  Resolved: 11/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Duplicate** | | | | | is duplicated by | [~~UNITSOFMEASUREMENT-112~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-112) | Refactor "internal" elements of RI fo... | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-86~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-86) | Getting rid of java.util.Locale and R... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-120~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-120) | Sync Unit Systems with oBIX | Resolved | | is related to | [~~UNITSOFMEASUREMENT-103~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) | Look into Unicode CLDR | Resolved | |  |
| **Tags:** | format, i18n/l10n |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| UCUMFormat and supporting helper classes (mostly in format.internal) shall be refactored into a locale-neutral format like SimpleUnitFormat. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Jan/15 ] |
| Getting this format Locale and ResourceBundle agnostic should help further ME integration |

|  |  |
| --- | --- |
| **[Remove UCUM from RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-114)** ([UNITSOFMEASUREMENT-114](https://java.net/jira/browse/UNITSOFMEASUREMENT-114)) [UNITSOFMEASUREMENT-115] [Remove UCUM system](https://java.net/jira/browse/UNITSOFMEASUREMENT-115) Created: 22/Jan/15  Updated: 22/Jan/15  Resolved: 22/Jan/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | unit, utils |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Remove the UCUM class |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-114] [Remove UCUM from RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-114) Created: 22/Jan/15  Updated: 09/Jun/15  Resolved: 09/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-109~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-109) | Clarify behavior of getSymbol() by a ... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-107~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-107) | Move some elements to a package "spi" | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-115](https://java.net/jira/browse/UNITSOFMEASUREMENT-115) | [Remove UCUM system](https://java.net/jira/browse/UNITSOFMEASUREMENT-115) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-116](https://java.net/jira/browse/UNITSOFMEASUREMENT-116) | [Refactor UCUMFormat into a locale neu...](https://java.net/jira/browse/UNITSOFMEASUREMENT-116) | Sub-task | Resolved | keilw | |  |
| **Tags:** | refactoring |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As discussed, UCUM support shall be removed from the RI to solve discrepancies between some elements in the UCUM system and more importantly get the size of the RI down.  The SE port or other implementations should cover that. If UCUM support for ME Embedded is desired, it may done so but it does not have to be part of the RI. This task will likely be split into smaller sub-tasks. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-113] [Set up CI for key repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-113) Created: 19/Jan/15  Updated: 12/Jun/15  Resolved: 12/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200), [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | automation, testing |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At least the most important artifacts other than the API (which is already built on GeoAPI) should be compiled and monitored by a CI server.  Options/evaluated are:   * Travis-CI * drone.io |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 12/Jun/15 ] |
| Single module projects like API, RI or SE-port mostly use drone.io. Due to insufficient Maven multi-project support (at least build results always show "failed" despite everything built properly) all other projects currently use Circle-CI, see <https://github.com/unitsofmeasurement/uom-lib> |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-112] [Refactor "internal" elements of RI format](https://java.net/jira/browse/UNITSOFMEASUREMENT-112) Created: 19/Jan/15  Updated: 27/Jan/15  Resolved: 27/Jan/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Duplicate | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Duplicate** | | | | | duplicates | [~~UNITSOFMEASUREMENT-116~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-116) | Refactor UCUMFormat into a locale neu... | Resolved | |  |
| **Tags:** | format, refactor |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| To distinguish from a dedicated UCUM parser against a **ucum-essense.xml** file some classes in "\*.format.internal" should be renamed see <https://github.com/unitsofmeasurement/uom-se/issues/55> <https://github.com/unitsofmeasurement/uom-se/issues/56> |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 27/Jan/15 ] |
| This partly duplicates the removal of UCUMFormat. Most of becomes irrelevant if the corresponding format is removed. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-111] [Test-related items like TCK or JUnit Tests, etc.](https://java.net/jira/browse/UNITSOFMEASUREMENT-111) Created: 19/Jan/15  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.0](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16917) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Epic | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Name:** | Test |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-110] [Presentation and UI aspects including i18n, Formatting, Parsing](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) Created: 18/Jan/15  Updated: 22/Oct/16  Resolved: 22/Oct/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.x](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17750) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Epic | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Name:** | View |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-109] [Clarify behavior of getSymbol() by a TransformedUnit](https://java.net/jira/browse/UNITSOFMEASUREMENT-109) Created: 17/Jan/15  Updated: 20/Oct/15  Resolved: 20/Oct/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-108~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-108) | getSymbol() and toString() differ for... | Closed | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-114~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-114) | Remove UCUM from RI | Resolved | | is related to | [~~UNITSOFMEASUREMENT-103~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) | Look into Unicode CLDR | Resolved | |  |
| **Tags:** | i18n/l10n, string, ui |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At least the concrete class TransformedUnit receives its symbol (returned by getSymbol() via the parent unit from which it is derived. To a certain extent this may affect every **derived unit**. While formatting and output via toString() works as designed for the cases tested and other output, e.g. via UnitFormat also have the same result, the getSymbol() method of derived types like GRAM, POUND etc. all show the symbol of the **base unit** being KILOGRAM in this case, but similar examples for other quanties should produce a similar outcome.  Some users feel this was a bug. As of now TransformedUnit is designed like that so it "workes as designed". Either we may want to tweak the behavior or clarify in JavaDoc and where necessary in the spec, why this method returns e.g. "kg" while toString() returns "g", "oz", "lb" or similar. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Jan/15 ] |
| The API JavaDoc <http://www.unitsofmeasurement.org/jsr-363/apidocs/javax/measure/Unit.html#getSymbol(>) suggests, null would be returned in cases like product of units. Which is why either JavaDoc of concrete classes need further clarification or the implementation should change to reflect that. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Oct/15 ] |
| As of now, most issues related to a TransformedUnit not holding a symbol can be solved by using UnitFormat.label() |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-108] [getSymbol() and toString() differ for several Mass units](https://java.net/jira/browse/UNITSOFMEASUREMENT-108) Created: 15/Jan/15  Updated: 06/Jul/15  Resolved: 19/Feb/15 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Duplicate | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-109~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-109) | Clarify behavior of getSymbol() by a ... | Resolved | | **Duplicate** | | | | | is duplicated by | [~~UNITSOFMEASUREMENT-139~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-139) | Special GRAM format problem with sub-... | Resolved | |  |
| **Tags:** | i18n, symbol |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As UnitsTest.testGetSymbol() shows, the result of getSymbol() differs from toString() which returns the expected value. Most Mass units have only the SI base symbol "kg" rather than their expected symbol. The string representation however is correct.  This also seems very similar for the SE port |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Feb/15 ] |
| This was more generalized by #109 |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Feb/15 ] |
| Closed as duplicate |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-107] [Move some elements to a package "spi"](https://java.net/jira/browse/UNITSOFMEASUREMENT-107) Created: 12/Jan/15  Updated: 19/Apr/15  Resolved: 19/Apr/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-114~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-114) | Remove UCUM from RI | Resolved | |  |
| **Tags:** | naming, package |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to the API some elements currently in "util" like SystemOfUnits implementations (SI, UCUM,...) could be moved into a new package "spi". Making the package name more consistent with the API. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-106] [Try to declare members of Units as Unit, not concrete classes](https://java.net/jira/browse/UNITSOFMEASUREMENT-106) Created: 12/Jan/15  Updated: 03/Nov/15  Resolved: 03/Nov/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-98~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-98) | Consider providing the "Units" class ... | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-105~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-105) | KILO(GRAM) misformatted as Mg | Resolved | |  |
| **Tags:** | design, implementation |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Some member variables in Units are declared as concrete types like BaseUnit at the moment. This not only affected formatting (see bug reference) it makes the idea of an API level class backed by SPI hard. Where nothing is broken the external declaration of all member variables should be changed to Unit<Q> |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-105] [KILO(GRAM) misformatted as Mg](https://java.net/jira/browse/UNITSOFMEASUREMENT-105) Created: 12/Jan/15  Updated: 18/Jan/15  Resolved: 12/Jan/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-106~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-106) | Try to declare members of Units as Un... | Resolved | |  |
| **Tags:** | format, i18n |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| See unit-ri tec.units.ri.util.PrefixTest testKg(tec.units.ri.util.PrefixTest) org.junit.ComparisonFailure: expected:<[k]g> but was:<[M]g>  uom-se does not show a similar problem when running the same unit test or demo code. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-104] [UCUM prefix unit misspelled TRIILLIONS](https://java.net/jira/browse/UNITSOFMEASUREMENT-104) Created: 11/Jan/15  Updated: 18/Jan/15  Resolved: 12/Jan/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=17107) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | typo |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The UCUM system class contains a typo,  TRIILLIONS should be spelled TRILLIONS |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-103] [Look into Unicode CLDR](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) Created: 24/Dec/14  Updated: 10/Aug/15  Resolved: 10/Aug/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16886) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-86~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-86) | Getting rid of java.util.Locale and R... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-100~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) | Define our criterion for inclusion / ... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-109~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-109) | Clarify behavior of getSymbol() by a ... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-116~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-116) | Refactor UCUMFormat into a locale neu... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-120~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-120) | Sync Unit Systems with oBIX | Resolved | | is related to | [~~UNITSOFMEASUREMENT-121~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) | Introduce "Pluggable" Unit Systems an... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-100~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) | Define our criterion for inclusion / ... | Resolved | | relates to | [~~UNITSOFMEASUREMENT-126~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-126) | Should we rename SI to Metric? | Resolved | |  |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| OpenJDK declared at least 2 JEPs related to CLDR: This one's alread final (as of Java 8) <http://openjdk.java.net/jeps/127> This one's a draft for Java 9 <http://openjdk.java.net/jeps/8043554>  The whole CLDR is likely too big at least for the RI, but where selective access and usage of CLDR data was possible, it is clearly worth exploring. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Jan/15 ] |
| Using the CLDR might improve some of that although getSymbol() is supposed to be locale-neutral while toString() isn't necessarily. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Feb/15 ] |
| Will try to use some of CLDR especially where locale-neutral |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Mar/15 ] |
| Where units provided by CLDR are to be supported this may change the number of quanties (or unit "type" as CLDR/ICU calls them) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Aug/15 ] |
| Descoping this from RI (where the locale-neutral SimpleUnitFormat and similar classes should be the default)  See <https://github.com/unitsofmeasurement/uom-se/issues/92> |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-102] [Wrong conversion factor of RI UnitConverter](https://java.net/jira/browse/UNITSOFMEASUREMENT-102) Created: 23/Dec/14  Updated: 24/Dec/14  Resolved: 24/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16886) |
| **Fix Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17107) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Blocker |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | unplanned | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | conversion, converter |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As a side-effect when running a real life example derived from the Spec for the performance discussion I stumbled over a much more severe problem. A bug in the RI (the SE port though both are derived from the same codebase seems unaffected) causing UnitConverter to misbehave. For a given code  Unit<Length> sourceUnit = METRE;  Unit<Length> targetUnit = CENTI(METRE);  UnitConverter converter = sourceUnit.getConverterTo(targetUnit);  the RI returns a RationalConverter with a 1/8 ratio, thus 4 m would become just 32 cm!!!  The Quantity.to() conversion acts the same way, so the problem has a common cause and it does not matter, if you use Quantity or not, the result stays the same.  The SE port "uom-se" does it correctly.  Anybody who has a chance to look into that over the holidays, please do, it's the biggest issue at the moment. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| Starting to analyze the problem.  The necessary downsizing of RationalConverter from BigInteger to long seems a main part of the problem. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| OK, now I was able to narrow it down before UnitConverter as such was constructed in above example.  Unit<Length> targetUnit = CENTI(METRE);  creates a TransformedUnit with converter=RationalConverter(100,1) for SE, while the same for the RI says converter=RationalConverter(1,8). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| When downsizing the SIPrefix I converted the RationalConverter with BigInteger.TEN.pow(something) to a simple primitive arithmetic, but unfortunately Java isn't a functional language, so "10^3", etc. works perfectly fine, compiles and everything but it does some strange "bitwise shift" instead, probably why 10 turned into 8 instead of 100  About to fix it in the RI. We'll simply release a new version of the RI. If PMO can attach a new RI like "0.7.1" it'll also be addressed there before they put it online after Christmas. It is not an issue of the Spec or API. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Dec/14 ] |
| It's a little more complicated. Evidently, CLDC 8 did not get a Math.pow() method yet either (either for performance or other reasons, that seems among the classes unchanged from J2ME times) so the only alternative is to substitute it (similar to the gcd() method) or try using the explicit numbers. For 10^24 to 10^-24 that requires double as long is exceeded. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-101] [Create TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-101) Created: 20/Dec/14  Updated: 02/Jul/16  Resolved: 02/Jul/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16886) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-173~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-173) | Verify minimum Java version for TCK | Resolved | | is blocked by | [~~UNITSOFMEASUREMENT-176~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-176) | Consider dropping SPI profile from TCK | Resolved | | is blocked by | [~~UNITSOFMEASUREMENT-153~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-153) | Portability of TCK | Resolved | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-169~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-169) | Consider additional profile(s) for TCK | Resolved | | depends on | [~~UNITSOFMEASUREMENT-174~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-174) | Consider using Reflections.org for TCK | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-117](https://java.net/jira/browse/UNITSOFMEASUREMENT-117) | [Add Tests according to Spec Chapter 2](https://java.net/jira/browse/UNITSOFMEASUREMENT-117) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-118](https://java.net/jira/browse/UNITSOFMEASUREMENT-118) | [Add Tests according to Spec Chapter 4](https://java.net/jira/browse/UNITSOFMEASUREMENT-118) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-158](https://java.net/jira/browse/UNITSOFMEASUREMENT-158) | [Add Tests according to Spec Chapter 5](https://java.net/jira/browse/UNITSOFMEASUREMENT-158) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-159](https://java.net/jira/browse/UNITSOFMEASUREMENT-159) | [Move test-audit.xml to main/resources](https://java.net/jira/browse/UNITSOFMEASUREMENT-159) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-169](https://java.net/jira/browse/UNITSOFMEASUREMENT-169) | [Consider additional profile(s) for TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-169) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-173](https://java.net/jira/browse/UNITSOFMEASUREMENT-173) | [Verify minimum Java version for TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-173) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-174](https://java.net/jira/browse/UNITSOFMEASUREMENT-174) | [Consider using Reflections.org for TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-174) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-176](https://java.net/jira/browse/UNITSOFMEASUREMENT-176) | [Consider dropping SPI profile from TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-176) | Sub-task | Resolved | keilw | |  |
| **Tags:** | tck-red, test, testing |
| **Epic Link:** | [Test](https://java.net/jira/browse/UNITSOFMEASUREMENT-111) |
| **Sprint:** | Q1/15, Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Following Early Draft, for any subsequent stage we shall start a TCK. While using TestNG, not JUnit the TCK of JSR 354 looks like a good inspiration for those who want to learn what it shall look like: <https://github.com/JavaMoney/jsr354-tck> JSR 107 is also a (currently) standalone JSR: <https://github.com/jsr107/jsr107tck> but adds some complexity by testing against various containers like Weld, Guice, Spring,... Something we will face in a slightly different form, becaus at least on Java ME and SE Embedded implementations are expected to run and be tested. JSR 361 (MEEP 8) was the first offering Optionality, but except to those among us in the EC its TCK is not Open Source or publicly visible.  There should be sub-tasks, others are extremely welcome to help. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-100] [Define our criterion for inclusion / exclusion of quantity types](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) Created: 19/Dec/14  Updated: 02/Feb/16  Resolved: 20/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Critical |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |
| --- | --- |
| **Attachments:** | geoapi.xlsx     nist\_units.xlsx     quantities.pdf     quantities.xlsx |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-20~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-20) | Should we move Dimensionless into a (... | Resolved | | is blocked by | [~~UNITSOFMEASUREMENT-9~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-9) | Mass Density Unit | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-121~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) | Introduce "Pluggable" Unit Systems an... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-103~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) | Look into Unicode CLDR | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-181~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-181) | Refine quantities included in API | Resolved | | relates to | [~~UNITSOFMEASUREMENT-120~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-120) | Sync Unit Systems with oBIX | Resolved | | relates to | [~~UNITSOFMEASUREMENT-103~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) | Look into Unicode CLDR | Resolved | | relates to | [~~UNITSOFMEASUREMENT-127~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-127) | Simplify unit systems in RI | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-141](https://java.net/jira/browse/UNITSOFMEASUREMENT-141) | [Apply criterion for inclusion / exclu...](https://java.net/jira/browse/UNITSOFMEASUREMENT-141) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-142](https://java.net/jira/browse/UNITSOFMEASUREMENT-142) | [Document criterion for inclusion / ex...](https://java.net/jira/browse/UNITSOFMEASUREMENT-142) | Sub-task | Resolved | keilw | |  |
| **Tags:** | design, document, question, spec, vote |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The javax.measure.quantity package has a large number of quantity types, but the criterion for accepting or rejecting a quantity in this list is unclear. Given that the amount of possible quantities is very large and we do not want to include all of them, we should probably define our criterion. Mismatch between specification and API Maybe as an effect of the criterion not being explicitly defined, the table provided in the "Supported quantities" section of the specification is missing the following types compared to the current Javadoc / Spec:   * Action * AngularAcceleration * AngularSpeed * Dimensionless * DynamicViscosity * ElectricPermittivity * Information * InformationRate * IonizingRadiation * KinematicViscosity * Luminance * MagneticFieldStrenght * MagneticPermeability * MassFlowRate * Torque * VolumetricDensity * VolumetricFlowRate * WaveNumber |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| There are 2 chapters, "Supported Units" refers to BIPM and others, but I guess we should probably merge these two chapters in the Spec and add further documentation into a "package-info" of the "quantity" bundle. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 22/Dec/14 ] |
| Actually this task is also for checking if the current quantity types comply with the chosen definition. I though that the JSR-275 specification was compliant with BIPM, and we see that there is a mismatch between the specification and JSR-363. I think we should add a column in the table that document the provenance of each quantity type. Ideally, that provenance should be in the javadoc too. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Dec/14 ] |
| This part is based on the JSR-275 contributions to the spec. There is nothing that changed with 363 except a few names like Speed vs. Velocity or Radioactivity. If a mismatch existed, it was always there. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 22/Dec/14 ] |
| This is not what I mean. A lot of new types have been added compared to JSR-275, and I don't know on which criterion those new types were added (e.g. from which standards they come from). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Dec/14 ] |
| Well they are practically identical to: <http://www.unitsofmeasurement.org/apidocs/index.html>  So many that were not in 275 have been part of the 0.6 API that's out and used.  This is actually the most interesting source: <http://en.wikipedia.org/wiki/Proposed_redefinition_of_SI_base_units>  E.g. that's where Action came from. We can't wait for the 26th 'General Conference on Weights and Measures' in 2018 since at least Java 9 (or even 10 are likely to be final before that, but it seems a good idea to refer to this Wikipedia page and the 2011 proposal (which was after JSR 275 ended, so a few of the Unit-API 0.6 contents must be based on that already)  If the actual submission in 2018 brings critical changes, we may take it into consideration with either a MR or V 1.x/2.x of the Spec by then. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 22/Dec/14 ] |
| The above link is interesting, but it is about improving the definition of the current 7 base units - it does not introduce any new quantity. Maybe the link that you wanted to paste was an other one? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Dec/14 ] |
| No it was correct, as some of the contents like Action are related to <http://en.wikipedia.org/wiki/Proposed_redefinition_of_SI_base_units#Physical_constants_used_in_the_new_definitions_and_other_physical_constants> E.g. it was meant to define the Planck Constant. Boltzmann is also there  Aside from that a significant number of quantities that do not directly refer to a constant like those mentioned were added to represent all types in the UCUM system. There Bit, Byte, etc. exist, while e.g. the US system in the RI seems to exceed some of the typical "US Customary" units, e.g. OCTET (a unit of Information) does not seem to show up in "traditional" US Customary, but UCUM contains it all: <http://unitsofmeasure.org/ucum.html#section-Tables-of-Terminal-Symbols>  There you'll also find the source for the Action quantity and almost every other (e.g. Information as short form for "Amount of Information") To cover UCUM in a strongly typed manner, all of these seem to make sense. Usually the UCUM doc also refers to where they came from. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 23/Dec/14 ] |
| In my understanding the "Physical constants used in the new definitions" link does not add new quantities. It explains the relationship between physical constants and the new definitions of the 7 base units. So is there somewhere a link about new quantities that would be added by the International Committee for Weights and Measures (CIPM)? I still don't see any.  UCUM can help us, but I don't think that it can be cited as an authoritative source. Recognized standard organizations are CIPM and the like, not UCUM (even if widely used). Governments for example, in my understanding, would not officially accept UCUM as a source of standards (even if some departments may adopt is unofficially). They tend to accept only international standards of recognized bodies like ISO. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Feb/15 ] |
| If the sources for unit systems was pluggable, that could mean additional quantities. If type-safety is required, this means additional modules defining such quantities will be needed by certain "unit sources", as the most common denominator, javax.measure.quantity should contain e.g. the key units in the SI system. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Apr/15 ] |
| CLDR/oBIX/JSR363 (UCUM-inspired) comparison |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Apr/15 ] |
| PDF export of the comparison sheet |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Apr/15 ] |
| If we use the **CLDR** definitions and (where CLDR was missing any of them) units listed here: <http://www.npl.co.uk/reference/measurement-units/> we might reduce the number of quantity types in the API. Leaving others to implementations (like uom-se) or library modules.  WDYT? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Apr/15 ] |
| Here's a selection by **NIST** structured into nice 4 table groups: <http://physics.nist.gov/cuu/Units/units.html> A more detailed overview can be found in this 2008 publication: <http://physics.nist.gov/Pubs/SP330/sp330.pdf> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/Apr/15 ] |
| Table of types used by the **GeoAPI V3** standard: <http://www.opengeospatial.org/standards/geoapi> Except for AngularSpeed they all match the scope mentioned here, but to provide backward-compatibility with JSR-275 and existing GeoAPI standards, we should support these in the "quantity" package of the API rather than forcing additional dependencies. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Apr/15 ] |
| One more data source and unit catalog came to my attention. Especially Geo and Climate projects refer to **UDUNITS** as alternative to UCUM. Essentially it is a close counterpart to **UCAR** Units written in C. What makes it interesting are several XML files, similar to oBIX or UCUM, see <https://github.com/Unidata/UDUNITS-2/tree/master/lib>  As UCAR lacks **typesafe** generic quantities the quantity is not mentioned but <https://github.com/Unidata/UDUNITS-2/blob/master/lib/udunits2-derived.xml> in addition to <https://github.com/Unidata/UDUNITS-2/blob/master/lib/udunits2-base.xml> may help gather the desired quantities for JSR 363. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 20/Apr/15 ] |
| Hi.  I can't open the NIST pages; don't know what's the problem.  I think the issue is more of scope than of size. As Jean-Marie said before, if there's a NIST standard, let's stick to it. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Apr/15 ] |
| Maybe it's some proxy or firewall issue...? The thing with "standard" is, even Table 2 of the mentioned NIST site and PDF document (guess you can't open that either?) says "Examples of SI derived units". So it does not say "definitive list", but the 3rd table calls them "named units", so that is sort of a definition by NIST.  There are one or two we should try to include for GeoAPI compatibility, as mentioned, otherwise I'd say we have a consensus on all 7 base quantities from table 1. All "named units/quantities" from table 3.  About table 2, allow me to list only the more questionable or seemingly "exotic" ones, and please +1 for each that you think the core API should contain (mainly the Quantity)  Quantity; Unit; Symbol  wave number; reciprocal meter; m-1  mass density; kilogram per cubic meter; kg/m3  specific volume; cubic meter per kilogram; m3/kg  current density; ampere per square meter; A/m2  magnetic field strength; ampere per meter; A/m  amount-of-substance concentration; mole per cubic meter; mol/m3  luminance; candela per square meter; cd/m2  mass fraction; kilogram per kilogram, which may be represented by the number 1; kg/kg = 1  Thanks, Werner |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/May/15 ] |
| 3 Tables of NIST Units compared to quantity types in API |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/May/15 ] |
| Updated NIST Units compared with API |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Jun/15 ] |
| While quantities to be pruned and removed from the API get a @Deprecated JavaDoc tag pointing to this ticket, I started using some new JavaDoc tags from Java 8 (see <http://www.javacodegeeks.com/2015/01/new-javadoc-tags-apinote-implspec-and-implnote.html>) for the others, mainly   * @apiNote SI base unit * @apiNote SI derived unit with special names/symbols * @apiNote SI derived unit   Except for Dimensionless which is a special case (not a real base unit though essential) all quantity types should carry this meta-information. It is not rendered unless you run JavaDoc for SE 8 and above, but it does not hurt Java 7 or ME 8 since those are just JavaDoc annotations. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-99] [Rename javax.measure.spi.\*Service as \*Provider](https://java.net/jira/browse/UNITSOFMEASUREMENT-99) Created: 19/Dec/14  Updated: 23/Jul/15  Resolved: 23/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-128~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-128) | Add ServiceProvider interface | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-122~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-122) | Should we rename SystemOfUnits? | Closed | |  |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The javax.measure.spi package defines the following interfaces:   * SystemOfUnitsService * UnitFormatService   However the usage in the JDK seems to use the Provider suffix instead than Service, as we can find in the following packages:   * java.nio.channels.spi * java.nio.charset.spi * java.nio.file.spi * java.text.spi * java.util.spi * javax.sound.midi.spi * javax.sound.sampled.spi * javax.sql.rowset.spi * javax.xml.ws.spi   Exceptions to the "Provider" suffix usage are below (but none of them use "Service"):   * javax.imageio.spi * javax.naming.spi * javax.security.auth.spi   I suggest to comply to JDK usage are replace "Service" by "Provider" in above-cited interface names. Note on semantic Note that "Service" and "Provider" may not have the same meaning. Maybe we can see a service as something like a "Web Service": we send a request to the service, it performs some work, we get an answer, then we send a new request based on user action or previous request results, and so on. The Fowler's description of "Micro Service" seems to be similar.  By contrast, a provider seems to be a "one shut" action, like a connection to a database: we get a "connection" (in our case, just a reference to a Java object) to a service. Then, we can forget about the provider.  Indeed, "SPI" in the package name stands for "Service Provider Interface": both "Service" and "Provider" are in the name. Providers provide services. In JSR-363, I think that UnitFormat can be seen as a kind of service. Consequently UnitFormatProvider would be the interface that give me UnitFormat services. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| While odd I don't think OSGi (which so far is the more common usage for this API and its precursors) would die if we had to implement a UnitFormatProvider either as UnitFormatProviderImpl or (that might even be an appealing side-effect ) UnitFormatService in an implementing framework.  One caveat is, that Java ME 8 puts a very strong emphasis on "service-enabled", see <http://www.oracle.com/technetwork/java/embedded/overview/javame/java-embedded-system-requirements-359229.html> and especially on the ME side, the term "Service" is almost more common than on the SE/EE side, e.g. <http://docs.oracle.com/javame/config/cldc/opt-pkgs/api/security/satsa-api/jsr177/javax/microedition/securityservice/package-frame.html> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Jul/15 ] |
| See <https://java.net/jira/browse/UNITSOFMEASUREMENT-128> As we introduced a ServiceProvider what it provides should be called \*Service in most cases. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-98] [Consider providing the "Units" class in the API](https://java.net/jira/browse/UNITSOFMEASUREMENT-98) Created: 19/Dec/14  Updated: 23/Feb/16  Resolved: 23/Feb/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Trivial |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-145~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-145) | Move Bootstrap mechanism into API | Resolved | | is blocked by | [~~UNITSOFMEASUREMENT-106~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-106) | Try to declare members of Units as Un... | Resolved | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-96~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-96) | Clarify if the optionality of quantit... | Resolved | | **Duplicate** | | | | | duplicates | [~~UNITSOFMEASUREMENT-56~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-56) | Could we move AbstractQuantity into A... | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-121~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) | Introduce "Pluggable" Unit Systems an... | Resolved | |  |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The Units class is currently provided in the reference implementation. It is technically possible to provide that class in the API part. However there is open questions:   * Optionality:   + If this class is provided in javax.measure package, does it means that it can not be optional?   + Do vendors have to provide every fields (is it all or nothing)? * The Units class would fetch the implementation using java.util.ServiceLoader. But what should we do if the service loader find no implementation? Or find more than one implementation? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| Java 8 and Oracle/JCP are clear on optionality, so along the lines of other JSRs like CDI we are most likely to define it in the spi package, see the CDI class defined by CDI 1.1 and above (or if we don't mind extra complexity it could be its own package like system)  The dependency currently only spi depends on format aside from the core "root" package, would have to be further explained, at least the quantity package is a mandatory requirement for such SI class and until some "Stripped Implementation" guideline and approach is permitted, the class must contain all methods/constants/SI types we chose to expose. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Jan/15 ] |
| As "Stripped Implementations" are off the table, a package like spi for a possible SI class on API level sounds best, that's where underlying interfaces are also declared. There is a dependency on format already, hence, if SI or similar systems could be added to the API it would also require the optional{{quantity}} module as dependency.  Another alternative while adding further modules could be declaring a separate module and package like system or systems for such classes. It would make spi independent of quantity and require both spi and quantity aside from core types itself. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/May/15 ] |
| As we tend to reduce the number of quantities to a reasonable number (not every single one in scope for SI, but the most commonly used ones) there won't be a SI class in the RI any more soon. However, the planned replacement (a class called Units) and ideally also a similar factory Quantities could be explored to make them available in the API. (similar to JSR-354) |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 23/Feb/16 ] |
| I don't think this is necessary, and adds more complexity than value in my opinion! |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Feb/16 ] |
| Thanks for the input. I agree, and since we have a SystemOfUnitsService offered by Bootstrap (or a different name should we prefer) one can query all available unit systems, not just SI (which we only cover to some extent in the API/RI) but also UCUM, ISO80k or others. |

|  |  |
| --- | --- |
| **[Non-functional Requirements (mostly for RI)](https://java.net/jira/browse/UNITSOFMEASUREMENT-185)** ([UNITSOFMEASUREMENT-185](https://java.net/jira/browse/UNITSOFMEASUREMENT-185)) [UNITSOFMEASUREMENT-97] [Add a chapter about performance](https://java.net/jira/browse/UNITSOFMEASUREMENT-97) Created: 19/Dec/14  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-184~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-184) | Loss of precision by using double ins... | Resolved | | relates to | [~~UNITSOFMEASUREMENT-155~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-155) | Create separate RI documentation | Resolved | |  |
| **Sprint:** | Final Release, Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Performance allowed by the JSR-363 API may deserve a discussion on its own. I propose to add a section on this topic. Key elements are:   * Usage of Quantity will always be much slower than usage of double primitive type, for at least two reasons:   + Wrapping primitive type in an object ("value types" in a future Java version may help)   + Check for unit compatibility and conversions ("value types" will be of no help for that). * Users can store the data themselves in their own structure, and use UnitConverter themselves when needed.   + No wrapping of primitive types   + More important, if there is lot of values to convert between the same pair of units, users can get the UnitConverter only once (by comparison, Quantity would perform this very expensive operation for each individual values) * Users can get yet more performance by extracting the scale and offset factor from the UnitConverter when the later is linear, and integrating those factors in their processing chain (e.g. in an affine transform). |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Dec/14 ] |
| I am not sure, if we should spend too much time on that. Of course, JSR 310 is much slower than especially the good old java.util.Date or Calendar, but somewhat more intuitive (not everywhere ) usage by developers apeals to many despite the performance penalty. Similar with say JSR 354.  We can certainly try to explain how UnitConverter may help in certain niche cases where one would prefer primitive types over the more type consistent and functional friendly Quantity but we'd cut the tree we're just about to build if we tell everyone "go use primitive types, don't touch Quantity". It is clearly a "Non Functional" requirement and for a Functional JSR like this (along with others like 354, 310) aiming at new data types this is not a priority, at least where we can only trust in the underling JVM to be improved where necessary. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 22/Dec/14 ] |
| I don't think we would cut the tree. I think that numerical intensive applications will never accept to use Quantity. Myself I will not, except for metadata were performance is much less critical. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 22/Dec/14 ] |
| To be more specific, Quantity is really suitable only for some specific cases: when we have only one numerical values (not an array or matrix or tensor), and when memory usage and performance doesn't matter much (as in metadata). For other cases where there is many numerical values in the same units (e.g. a vector), then I think that usage of Quantity should be **discouraged**, and UnitConverter encouraged instead.  From this point of view, I do not think that the Range class is a good examples for JSR-363 users. I think that Range should be implemented with two double primitive values and a single Unit, not two Quantity fields. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Dec/14 ] |
| Seems we're moving in circles at least around the non-numerical values. The initial draft had a more abstract Quantity (then called Measurement) that was perfectly fine for an array, matrix or whatever (e.g. you could use a Java array rather than a single number) All of this was disregarded and is no longer part of the API, so either we made a mistake or the calculations with vectors and primitives are just a small niche?  Even Java 8 and other pieces of software still struggle with the duality of primitives vs. objects, but especially an increasing competition from pure OO languages like Scala, etc. make it evident, Java also wants to move further in that direction, see the "Value Object" proposal. For backward compatibility Java will likely keep primivite types for a long time, but we're not writting a JSR that still works with legacy J2ME like JSR 256 In theory see <https://docs.oracle.com/javase/8/docs/api/java/util/function/DoubleBinaryOperator.html> we could also provide a DoubleRange to the RI, but we're already flexible by allowing Number as to chose between heavy, but very precise type like BigDecimal vs. smaller like Double or Integer which pretty much wrap only the underlying primitive.  While the type hierarchy may be a bit weird at times, both OpenXC or Google Fit, etc. aim at "Real Time" data and none of them bother using primitives like double or long. It's Number everywhere. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Dec/14 ] |
| Actually quite bizarre (given ex Google employee Josh Bloch highly disregarded the use of float for Java) the core data type of Google Fit called Value offers 2 underlying numeric representations, one int the other float , nevertheless all of the API takes the wrapper type as arguments, never primitives directly. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 22/Dec/14 ] |
| The issue is not about primitives versus wrappers (while it still a significant issue I think). The issue is that when storing an array of Quantity instances (say 100 for this example) we got the following inconvenients:   * We store 100 (Number, Unit) tupples instead of 100 numbers (possibly as primitives) + 1 Unit. * Having 100 Unit values, we need to get the UnitConverter 100 times, each time for converting only one Number. This is **much** slower than getting the UnitConverter only once, then applying it to 100 numbers. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Dec/14 ] |
| We could offer it as sort of a recommendation with "Mass data" or "Big Data". JSR 354 has some sort of benchmarks to compare e.g. the BigDecimal and primitive based implementations (the latter called FastMoney) We had something you wrote back in the old days of JSR 275, maybe something similar could accompany the RI again, but that's more to compare DoubleQuantity and DecimalQuantity.  As of now, the API no longer supports non-numeric quantities or any alternate ones. Only in the enum implementation there is an example to combine e.g. textual (enum-based) values with the unit via a UnitSupplier but all of this was removed from the API, so it only offers number quantities now The separate functional interfaces were not there to be "complicated", but to provide flexibility for alternate data structures where required  As we decided to get rid of it, there's little point to write about a structure with a unit and an array of numbers in the Spec because the Spec simply does not support it any more. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 23/Dec/14 ] |
| The point is to explain to users how to use JSR-363 efficiently in their own data structure, not to provide new structures. We removed non-scalar structure from JSR-363 because we can not cover all needs adequately. It does not mean that we can not help users to handle units in their own data structures. Why would you want to force all developers to use our own data structure?  The guideline is: if you have a collection of values in the same unit, do not use Quantity. Instead, define your own structure as efficient as you wish, and use UnitConverter for unit conversions. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| That's about as saying if you have a big set of data, don't use java.util.Date (not to mention the new java.time elements that carry a huge payload, usually even collections of "fields" for year, month, day,...) but store the date as long.  We should simply phrase something adequate into the chapter about "Unit Conversion" and further elaborate about a "Mass data" example similar to the one already given:  Unit<Length> sourceUnit = METRE;  Unit<Length> targetUnit = CENTI(METRE);  UnitConverter converter = sourceUnit.getConverterTo(targetUnit);  double length1 = 4.0;  double length2 = 6.0;  result1 = converter.convert(length1);  result2 = converter.convert(length2);  Either right there in a separate paragraph, or somewhere else (e.g. "Examples") but not try a separate "Performance" stunt. Wherever Java offers primitive types they generally require less memory or time to deal with, but this small running variation of our spec example shows why Quantity is safer and we should not discourage its use (we may offer a chapter for performance or mass data usage, but with the caveat, that primitives are not unit-consistent, and the compiler won't tell mistakes there )  Unit<Length> sourceUnit = METRE;  Unit<Length> targetUnit = CENTI(METRE);  UnitConverter converter = sourceUnit.getConverterTo(targetUnit);  double length1 = 4.0;  double length2 = 6.0;  double result1 = converter.convert(length1);  double result2 = converter.convert(length2);  System.out.println(result1);  System.out.println(result2);  Quantity<Length> quantLength1 = Quantities.getQuantity(4.0, sourceUnit);  Quantity<Length> quantLength2 = Quantities.getQuantity(6.0, targetUnit);  Quantity<Length> quantResult1 = quantLength1.to(targetUnit);  System.out.println(quantResult1);    double mass1 = 5.0;  double result3 = converter.convert(mass1); // does compile  System.out.println(result3);  Quantity<Mass> quantMass1 = Quantities.getQuantity(5.0, KILOGRAM);  quantMass1.to(targetUnit); // won't compile |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 23/Dec/14 ] |
| Indeed, when having a big set of data we do not use java.util.Date. We use long or double. I would argue that this is very obvious to anyone working with big (or even medium) data. The NetCDF library is one example among other.  In addition of being a waste of memory and CPU, using Quantity for big data add more confusion than safety. This is because you add an unnecessary degree of freedom: each individual value could theoretically have its own unit, different than other values. We usually really want to enforce a uniform unit for the whole vector or matrix, and a good way to do that is to ensure that there is only one Unit reference for the whole data set, not one Unit reference per value. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| While calling it "format" rather than "unit", the Value of Google Fit also carries a type information in every single data point. The DataPoint has an explicit type, and format is supposed to match. So there's a deliberate redundance it seems Since we don't define a UnitSupplier all someone with special needs could do is choose between a Google like approach where they build arrays of Quantity and in theory each item may have a different type (say a sensor device sends values for heart rate, temperature or others, then each must be strongly-typed, even if they may come in one stream or set or define their own completely free structure with primitive types and if they want a Unit on top, but in the absense of API interfaces other than Unit itself this is completely up to each solution. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 23/Dec/14 ] |
| I do not know Google Fit. If it is like a spreadsheet (OpenOffice Calc, Microsoft Excel), the later choose by design to allow each cell to have its own properties (color, font, etc.). Note that 65536 values (the spreadsheets limit until a few years ago) is not big data, and even at this amount spreadsheets were used to become very slow. At the same time, meteorologists were used to work with billion of values.  One can do the test: create an array of 1 billion double values in a primitive array and watch the performance of converting them with a single UnitConverter. Then compare with 1 billion Quantity instances and the time needed to convert them. Keep in mind that even 1 billion values is "small" by today meteorology / oceanography / Earth observation (and maybe Twitter, etc.) standards.  We excluded an API other than Quantity on intend. We let users define their own structure on intend. We provide a UnitConverter interface exactly for that purpose. What I'm suggesting is that we should explain to users which disaster they will be faced if they try to store 1 billion Quantity instances, and what is the correct way to use JSR-363 in this scenario: have a single Unit and use UnitConverter. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| No, Google Fit runs on Android devices gathering body measurements. So it may not be "Huge Data" in this case but it is meant for tiny devices. Of course they normally have more memory than most huge app servers did a fe years ago. Another reason why we could try to "assist" people how they could use UnitConverter, etc. in a "Big Data" like scenario, but there are so many use cases we considered out of scope, e.g. persistence (JPA, JSON,...) where one also has to chose what to use, so I don't think we should elaborate that too deeply.  There are more critical issues, see the RI problem with UnitConverter I just filed |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 23/Dec/14 ] |
| Scientists usually store there data as float. So let compare an array of Quantity with an array of float on a machine using 64 bits references:   * The array of Quantity references: equivalent to 2 arrays of float primitive values. * For each Quantity instance:   + Object header: equivalent to 3 float on Hotspot server   + The double value: equivalent to 2 float   + The reference to the Unit object: equivalent to 2 float   So an array of Quantity instances consume at least 9 time the memory of an array of float in 64-bits addressing. With 32-bits references instead than 64 bits, it still consume 7 time more memory. This does not include the huge pressure on the garbage collector for those billions on instances. Even if we had "value types", it would still consume 3 or 4 times (depending if 32 or 64-bits addressing) the memory of an array of float.  This memory issue is in addition to the above-cited performance issue. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| That may certainly be the case, but it applies to BigDecimal vs. double or any other POJO, just as well, so it is not specific to this JSR (we are not changing the underlying JVM or platform here )  Probably explains why Google used float for some Android apps like Fitness. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 23/Dec/14 ] |
| I agree, but this is why I think we need to explain that in a performance chapter:   * Use Quantity when there is few values, or when each value can have an unit independent of other values. * Use your own efficient data structure with a single (or few) Unit and UnitConverter for big data. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| I would probably call it differently like "Mass Operations" or similar, not even sure, where in the Spec (it is certainly nothing that belongs to the main chapter, we must create a TCK very close to the Spec, see JSR 354 or CDI, etc. )  There is not a Spec I know that would say "Don't use that because the JVM cannot perform so well with it", nor are JSRs obliged to run performance benchmarks as part of the mandatory tests, but if you really insist on that please consider creating some sort of benchmark in the RI, then we can see hwo to refer to it... |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 23/Dec/14 ] |
| It is not about "the JVM cannot perform well with it" (or not completely)! Please re-read my [comment posted yesterday](https://java.net/jira/browse/UNITSOFMEASUREMENT-97?focusedCommentId=382527&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-382527). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| In any case, we have so many mandatory tasks like TCK and recently not many EG members really contribute, that this is "Nice to have" but neither mandatory nor important. Please point to any JSR where this was elaborated?  Normally projects implementing it deal with that, so aside from maybe a side note in a chapter about "Using UnitConverter" it would be contraproductive and also alienate those who review and vote on JSRs. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 23/Dec/14 ] |
| I am in several JSRs also as EG Member, so taking JMS 2.x as an example, the final JMS 2 spec says something like  "There is typically a performance/reliability trade-off implied by this choice." under chapter 7.7. Or 12.4.5. Restrictions on use of injected JMSContext objects "This means that they will all use the same connection. This will reduce the resource usage of the application and improve performance." Along those lines, maybe in the chapter on Unit Conversion there could be similar advise, but neither JMS nor othr JSRs has a dedicated "Performance" chapter, so we should handle this in a similar way.  The performance/reliability trade-off is pretty similar, as you can do "wrong stuff" like converting double values actually holding a Mass to Length and similar, only inside a structure like Quantity the compiler will prevent it for all operations but the "mixed" ones (and even there we enforce at least that cast warning ) |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 23/Dec/14 ] |
| I do not mind if the text appears in a performance chapter or another chapter. I just think that we should explain somewhere to the users that there is two ways to use JSR-363:   * Quantity * Your own data structure + 1 Unit + using UnitConverter   and explain the tradeoff between the two, so users can chose what fit best their needs. I'm mostly neutral about in which chapter is should appear. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Jan/15 ] |
| Actually the way JSR 354 handled this seems the best approach for 363, too  2.2.7. Performance Aspects of a separate **User Guide** document delivered in Asciidoc with the RI: <https://github.com/JavaMoney/jsr354-ri/blob/master/src/main/asciidoc/userguide.adoc#214-registering-additional-currency-units>  It was not included in its spec, so we should try a similar approach, regardless where we end up putting a user guide or other appropriate document. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Nov/15 ] |
| As we now got a separate RI User guide for JSR 363 and at least the original Benchmark from JSR 275 (will try to incorperate it somewhere, most likely uom-demos as it is a console app, not unit test) also refers to elements now mostly in implementations, I tend to see that as RI-relevant rather than for the Spec itself. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Mar/16 ] |
| As for Precision I guess both make a good chapter of the RI guide. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 28/Sep/16 ] |
| Moved to RI User Guide, so not subject to the RI or Spec |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-96] [Clarify if the optionality of quantity package is all or nothing](https://java.net/jira/browse/UNITSOFMEASUREMENT-96) Created: 19/Dec/14  Updated: 19/Jan/15  Resolved: 19/Jan/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-23~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-23) | Explore Stripped Implementations | Closed | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-98~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-98) | Consider providing the "Units" class ... | Resolved | |  |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The javax.measure.quantity package is optional. But it is not clear if this is a "all or nothing" optionality. We would like to allow vendor to provide a quantity package with only the quantity types they need, but we are not sure if we are allowed to do that or not.  Whether the optionality is "all or nothing" or if the vendor is allowed to pick some types should be said in the specification for avoiding confusion. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| The only way to pick individual types inside an optional package would be this so-called "Stripped" or "Application-specific" implementation. As of now Oracle Legal and other members of the JCP EC (IP WG) are stalled, so no solution is to be expected in the near future. Until then it's "All or None" |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| See the linked/blocking issue, no vendor is allowed to "crop" all quantities but say Length from the javax.measure.quantity package. They may not have to use every single one, but must not tamper with the JAR (whether that's a huge monolithic "unit-api" jar or something like MEEP 8 with 3-4 smaller JARs, we shall explore in a separate task ) without breaking compatibility.  "Stripped Implementations" which though "javax.measure.quantity" is part of the API should apply to a package using the core Quantity type may solve this in future, but "Stripping" was explicitly purged from Java 8 by Oracle, so we won't see it in ME 8 for now either. Even all sorts of "Obfuscation" may reduce the size of the JAR or names within it when an app is deployed to a small device, but the rules for ME 8 Embedded also forbid "Stripping" thus removing types from CLDC 8, MEEP 8 or JSR 363 beyond not using an optional package is not possible. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Jan/15 ] |
| Based on a recent conversation and statements by Mark Reinhold at the EC F2F he made clear, Jigsaw just like OSGi won't offer a finer granularity than **package** or **bundle**. He also said, the "Stripped" implementation will likely never come also due to a soon released Java 9 where at least modularity and optionality similar to MEEP can be accomplished otherwise. It won't allow picking individual classes or methods (other than e.g. down-scaling from full SE to ME/CLDC in some cases, but that's only reasonable for platform parts) thus you need to use the **entire** "quantity" package with all interfaces or none of it and stick to your own custom quantities where required. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-95] [Revisit the need for the Parser interface](https://java.net/jira/browse/UNITSOFMEASUREMENT-95) Created: 19/Dec/14  Updated: 15/Nov/15  Resolved: 15/Nov/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Trivial |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This task is for gathering use case for the Parser interface, and eventually (depending on the use case) revisit if we should omit it.  One argument for Parser was to parse other kind of objects than Unit. However the ParserException extends MeasurementException, which make it (indirectly) JSR-363 specific. So I think we should gather use cases of someone wanting to parse an object related to measurements but still something else than a Unit. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| We'll explore that. Aside from an obvious UCUM parser (which does return Unit as of now) parsers returning at least a Quantity or other key types (there aren't too many in this small JSR ) are very likely. MeasurementException is the base type for all exceptions like DateTimeException is in java.time, which includes Dozens (or rather Hundreds ) of other types beside DateTime. In fact DateTimeParseException follows exactly the same pattern. We may rename the exception into something slightly more specific, but the same exception should apply to parsing Unit, Quantity or other possible results. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Apr/15 ] |
| There are concrete use-cases already in uom-tools. Thus closing, As the parser returns different values, not just Unit or Quantity a more specific name also doesn't seem to match. Should the naming really feel like an issue, we could revisit later in a new ticket. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 30/Sep/15 ] |
| Either in implementations (function package) or uom-lib-common we should declare this interface, but it is only used by UnitFormat so we can factor it out of the API. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/Nov/15 ] |
| Parser moved to RI for now |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-94] [Explain that formulas with units may be different than formulas with numerical values](https://java.net/jira/browse/UNITSOFMEASUREMENT-94) Created: 19/Dec/14  Updated: 28/Sep/16 | |
| **Status:** | Open |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.x](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17750) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Trivial |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) |
| **Resolution:** | Unresolved | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Maintenance Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Explain in the specification that while a sequence of arithmetic operations (addition, subtraction, multiplication, division, power, etc.) is usually the same for both the numerical values and the units of the result, there may be a trap in some cases. First draft of a proposed discussion in the spec Formulas with Quantity objects are not necessarily the same than formulas with double primitive types. For example consider the following operations:  A \* B \* C  If A, B and C are double primitive types and if C == 1, then the operations can by simplified as A \* B. However if A, B and C are rather Quantity objects, then the same simplification can be applied only if C == 1 and is dimensionless. Be aware that the last condition is not always true. For example we could have C = 1 (J/K)/m³. In such case the C constant can be dropped when performing calculations using the double primitive types, but not when performing calculation using Quantity objects. When converting a formula designed for the double primitive types to a formula using Quantity objects, the developer should keep in mind that the later formula may need more terms than the former one. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-93] [Clarify the meaning of "commensurable" in cases like Torque vs Energy](https://java.net/jira/browse/UNITSOFMEASUREMENT-93) Created: 19/Dec/14  Updated: 11/Sep/15  Resolved: 11/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The specification said:  Two units u1 and u2 are compatible if and only if u1.getDimension().equals(u2.getDimension()). First question What is the expected result of the following code? Shall c be the identity converter or shall the code throw an IncommensurableException?  Unit<Energy> energy = SI.JOULE;  Unit<Torque> torque = SI.NEWTON.multiply(SI.METRE).asType(Torque.class);  UnitConverter c = energy.getConverterToAny(torque);  If I interpret the specification literally, we would get an identity converter (no exception thrown). Is it what we want? Second question What is the expected result of the following code? Should the operation succeed or throw a ClassCastException?  Quantity<Energy> energy = factory.create(1, SI.JOULE);  Quantity<Torque> torque = energy.asType(Torque.class);  My feeling is that above Quantity.asType(Class) should throw an exception even if the units are commensurable. Do we agree on that?  If the above code throw an exception, should Unit.asType(Class) do the same? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| It's a Checked Exception (see where it's used, there was a reason for that so far) but wouldn't it be better to throw IncommensurableException? |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 19/Dec/14 ] |
| I suggest to discuss the exception class in an other thread or JIRA task, and interpret this task as "throw whatever exception this method is supposed to throw in case of incommensurable units or quantities". |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| That's fine, if the task is about being "commensurable" then just throwing a ClassCastException instead of the dedicated API one would put that exception into question |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 19/Dec/14 ] |
| This task is about clarifying what should be the behaviour of Unit.isCompatible, Unit.getConverterTo, Unit.asType and Quantity.asType when the two units/quantities have the same dimension but are still usually considered as different units. The Torque vs Energy case is maybe one of the best examples for working that out. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Dec/14 ] |
| You already mentioned both exceptions up there anyway. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/Sep/15 ] |
| As Torque is no longer in javax.measure.quantity, this does not belong to the specification (which at most may occasionally refer to RI elements, but not 3rd party extensions like SI or US)  It could be mentioned in the RI guide <https://github.com/unitsofmeasurement/unit-ri-userguide> or better flagged here if necessary: <https://github.com/unitsofmeasurement/si-units/issues> |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-92] [Replace [code] block by HTML in javadoc](https://java.net/jira/browse/UNITSOFMEASUREMENT-92) Created: 19/Dec/14  Updated: 30/Dec/15  Resolved: 30/Dec/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Final Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The API javadoc contains [code] blocks which were used to be processed by the colorapi plugin. This plugin is not (yet) on Maven Central. It may also be desirable to have the Javadoc properly formatted in IDE that do not know about custom plugins.  The proposal is to replace the following elements:  [code]  Java code line 1  Java code line 2  Java code line 3  [code]  by:  <blockquote>  <pre>Java code line 1  Java code line 2  Java code line 3</pre>  </blockquote>  Note that the <pre> HTML elements are on the same line than the first and last Java code line because all new-line inside <pre>...</pre> are significant. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-91] [Add isSystemAvailable(), getAvailableSystems() or similar to SystemOfUnitsService](https://java.net/jira/browse/UNITSOFMEASUREMENT-91) Created: 17/Dec/14  Updated: 03/Nov/15  Resolved: 03/Nov/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16886) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-122~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-122) | Should we rename SystemOfUnits? | Closed | | blocks | [~~UNITSOFMEASUREMENT-7~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-7) | Getting rid of java.lang.reflect | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-90~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-90) | UnitFormatService is missing a getAva... | Resolved | |  |
| **Tags:** | lookup, service |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| See linked issue for UnitFormatService this also applies here. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Mar/15 ] |
| If the name of the system changes, so should the name of the service/provider |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Nov/15 ] |
| getAvailableSystemsOfUnits() returns the available unit systems. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-90] [UnitFormatService is missing a getAvailableFormats() method](https://java.net/jira/browse/UNITSOFMEASUREMENT-90) Created: 17/Dec/14  Updated: 03/Nov/15  Resolved: 03/Nov/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-89~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-89) | SystemOfUnits.getUnits() needs to def... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-91~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-91) | Add isSystemAvailable(), getAvailable... | Resolved | |  |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The UnitFormatService interface has a getUnitFormat(String) method, but no way to know which String are legal values. I suggest to add the following methods:  /\*\*  \* Returns the formats recognized by {@link #getUnitFormat(String)}.  \*/  Set<String> getAvailableFormats();  I think that we should also standardize the spelling of a few common format (or maybe even provide static final String constants for them). Examples:   * If the provider supports the SI format, then the name of that format shall be spelled "SI" in upper-case. * If the provider supports the UCUM format, then the name of that format shall be spelled "UCUM" in upper-case. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Dec/14 ] |
| Looking at a very similar equivalent in JSR 354 (in fact there those things are often called \*Provider, but 354 is not even OSGi enabled AFAIK so it's less Service-affine than this JSR )  ExchangeRateProvider prov = MonetaryConversions .getExchangeRateProvider("ECB");  There is no convention that the provider has to be called "ECB" (though it's the internationally accepted acronym) and can't be "EZB" (for German speaking projects or clients) etc. same for "IMF", etc.  Even if common acronyms like "ISO", "UCUM" or "SI" may be widely accepted, at most we could offer a standard constant for something that is part of the API (though SI.class.getSimpleName() has usually been sufficient ) anything else is up to implementations and 3rd party libraries. If say anybody implements SystemOfUnits in a class Iso (which does stick to Oracle's most recommended naming pattern, even when common acronyms are involved, see IsoChronology etc. in JSR 310) or the same for IsoFormat and the implementation of the service tries a dynamic lookup of the class name, then "Iso" rather than "ISO" would be correct. I don't see a reason for overregulating this.  The actual subject of the ticket seems valid, similar to JSR 354  public static boolean isConversionAvailable(String currencyCode, String... providers)  we could have isFormatAvailable() or getAvailableFormats() (not ...Format()) in this service. Same for SystemOfUnits btw. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 17/Dec/14 ] |
| It is about regulating the String values, not the class name. Regulating the String values is a very common practice. See Charset, Locale, TimeZone, IIOMetadata in the JDK and many, many, many others.  Saying "if you supports SI, then it must be called "SI"" does not means that the implementor can not support other names too. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Dec/14 ] |
| For the actual format they usually do pattern matching, e.g. "yyyy.MM.dd G 'at' HH:mm:ss z" to get a date format. For UnitFormat itself or implementations (JDK SimpleDateFormat is also an extension/implementation of the fairly generic DateFormat or Format base classes) sure, we might want to think of patterns, similar in case of parsers and formatters for Quantity where applicable. However this is a simple lookup. Even Locale offering a multitude of accessors (some with 2 or 3 parameters, others with a single string like "en\_UK" vs. "en\_US" or just "en") can't be compared. I would only go as far as DateFormat: <http://docs.oracle.com/javase/8/docs/api/java/text/DateFormat.html> (not its subclasses like SimpleDateFormat) which is not very restrictive. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 17/Dec/14 ] |
| It is not just pattern. See Charset and IIOMetadata among others. Charset clearly standardize a few names like UTF-8. I don't see what is wrong about standardizing a few key names like SI and UCUM. Otherwise, how users are expected to get the UnitFormat instance for SI if there is nothing they can rely on? |
| Comment by [christianhansson](https://java.net/jira/secure/ViewProfile.jspa?name=christianhansson) [ 09/Apr/15 ] |
| Am i supposed to be on this project? seems like its a missclick |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 09/Apr/15 ] |
| You're not on the EG, but commenting or raising JIRA tickets is open to everyone |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Nov/15 ] |
| Similar to JSR 354, the service now got a method getAvailableFormatNames() returning a set of all available formats. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Nov/15 ] |
| In <https://java.net/jira/browse/UNITSOFMEASUREMENT-91> we currently return a collection of the actual systems. Whether or not that should be the same here, we may discuss. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-89] [SystemOfUnits.getUnits() needs to define which unit it returns](https://java.net/jira/browse/UNITSOFMEASUREMENT-89) Created: 16/Dec/14  Updated: 26/Oct/15  Resolved: 26/Oct/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Minor |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-90~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-90) | UnitFormatService is missing a getAva... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-124~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-124) | Introduce a Unit "DB" | Resolved | | is related to | [~~UNITSOFMEASUREMENT-88~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-88) | SystemOfUnits need to tell its base u... | Resolved | |  |
| **Tags:** | api, document, javadoc |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Current Javadoc of SystemOfUnits.getUnits() currently only said "Returns a read only view over the units defined in this system". This insufficient, since the set of units is potentially infinite.  I suggest to update the Javadoc with something like "Returns a read only view over units explicitly defined by this system. This include the base and derived units which are assigned a special name and symbol, like watt. This set does not include new units created by arithmetic or other operations". |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-88] [SystemOfUnits need to tell its base units](https://java.net/jira/browse/UNITSOFMEASUREMENT-88) Created: 16/Dec/14  Updated: 29/Jun/16  Resolved: 29/Jun/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Trivial |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Invalid | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-122~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-122) | Should we rename SystemOfUnits? | Closed | | **Duplicate** | | | | | is duplicated by | [~~UNITSOFMEASUREMENT-183~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-183) | getProductUnits should probably be re... | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-89~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-89) | SystemOfUnits.getUnits() needs to def... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-124~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-124) | Introduce a Unit "DB" | Resolved | |  |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The concept of "base units" is critical to the definition of a System of Units. Despite that, SystemOfUnits does not provide this information. I suggest to add the following method:  /\*\*  \* Returns the base units which by convention are considered  \* dimensionally independent of other base units.  \*  \* <p>The SI system defines only 7 base units (including metre, kilogram  \* and second) from which all other SI units are derived.  \* Other systems will have a different set of base units.</p>  \*/  Set<Unit<?>> getBaseUnits();  Note that I do **NOT** suggest to add a BaseUnit interface. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Dec/14 ] |
| Can you give a brief example what both the common non-SI systems (like US or Imperial though we also have a UCUM one thanks to JScience) would return there, and more importantly a domain-specific one like Health or Fitness?  If we just take a very brief Health system returning a patient's HeartRate (a custom quantity see "Heart of Glass") and Temperature would getBaseUnits() return Temperature while getUnits(( returns both? |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 16/Dec/14 ] |
| For an imperial system, base units would be foot, pound, second and Fahrenheit (maybe more, I'm not familiar enough with imperial system).  For the Health system mentioned above, I'm not sure that the question is appropriate: I didn't looked, but from the description HearthRate seems to be a quantity, not a unit. I presume that its unit would be Hertz, or something derived from Hertz. If the health system of units is SI, then there is nothing special to do.  Generally speaking, base units are subset of units: all elements returned by getBaseUnits() are also contained in getUnits(). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Dec/14 ] |
| Both Temperature and HeartRate would be quantities, of course. The unit (which in fact is a funny case, since you may find exactly the same in say a "Drum Computer" ) used by Heart of Glass is BPM.  Health and hundreds of other possible custom systems are exactly the reason why I want to clarify what sense it makes to add the method to SystemOfUnits on an API level.  E.g. if 90% or more of all implementations may end up returning null or an empty Set there could be little reason to add it at least in the API. In such case I'd prefer the RI |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 16/Dec/14 ] |
| We can explain in an other way: SystemOfUnits.getBaseUnits() return the set of legal keys of the Unit.getProductUnits(). In other words, the following assertion shall always hold:  assert systemOfUnits.getBaseUnit().containAll(unit.getProductUnit().keySet()); |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Dec/14 ] |
| P.s.: <p>The SI system defines only 7 base units (including metre, kilogram   * and second) from which all other SI units are derived. * Other systems will have a different set of base units.</p>   That's something to leave for an implementation of such method or (if we found, it makes no sense for the API level interface that would be the only place to declare it) RI level declaration.  The SystemOfUnit interface should not imply anything specific to just one or two implementations. If the method is so specific to SI it belongs to SI. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 16/Dec/14 ] |
| I'm not sure to understand the above comment. This method is not specific to SI: every systems of units have a set of base units. And the fact that SI has 7 base units is implementation-independent: all implementations of the SI system will have the same 7 base units. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Dec/14 ] |
| Then more importantly what would Health (<https://github.com/unitsofmeasurement/uom-lib/blob/master/domain/health-ri/src/main/java/tec/uom/lib/domain/health/ri/Health.java>) have? Right now it defines no Temperature so only domain-specific quantities Floor, HeartRate, Heartbeat (that's used to get a derived Hart rate from Heartbeat / Time) and Step are currently in that domain-specific unit system.  What would getBaseUnits() therefore return if it was in the API? |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 16/Dec/14 ] |
| The javadoc does not explain how the "health" system of units is defined. Do you have a reference?  Otherwise judging by the source code, the base units would be the "beat", "step", "floor" and at least "seconds", since the later is used in the definition of BPM. But I'm not really sure that this example is sane, since it mixes the SI system with a new one. Unless we consider it as an extension of SI. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Dec/14 ] |
| That's entirely up to the application or domain. Fitbit uses an enum called UnitSystem: <https://github.com/Fitbit/fitbit4j/blob/master/fitbit4j/src/main/java/com/fitbit/api/common/model/units/UnitSystem.java> but its domain model contains distance, floors or steps aside from other data points such as calories or volume of liquid.  They don't take steps into consideration, but what Fitbit calls their unit system is not based on a typesafe model like JSR 363, those units are merely used for a bit of UI presentation (and that is flawed, I tried it after entering say 300 ml in my account and it showed 300 fl oz!!! ) Just to show, that each device and vendors may have different units, let's assume a Fit or Health one tries to cover relevant quantities for Fitbit and a few other devices, then it could be Floor, Step, etc.  Let's just stick to our current demo, it does not matter as another vendor could define 2 others or stick only to HeartRate. If HealthX by a vendor "X" contains Floor, Step and HeartRate, HealthY by vendor "Y" "Calories" (sounds like Energy since that's measured in "J") and "Blood Pressure" (again likely Pressure) what would getBaseUnits() contain for both X and Y?  Trying to understand why e.g. Step or Floor would be considered "base units" in such system and the set would contain them instead of say just SECOND. That's what we'd have to make very clear in both spec and JavaDoc. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 16/Dec/14 ] |
| * Base unit are "atoms" from which all other units in a system are derived. * Derived units are products of base units raised to some power.   Those concepts are clearly defined in SI brochure, and not specific to SI. They are already used by JSR-363, in the Unit.getProductUnits() method, as the keys in the map are always base units, never anything else.  What confuse me with the Health example is that it doesn't seem to be a system of units. It seems to be a mix of 2 system of units: the SI one and a health one. Putting in the mix a HealthX and HealthY systems increase my confusion. To me, system of units are conceptually independent from each other (even if we commonly define conversion factors to SI). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Dec/14 ] |
| Well, then for all domains that wish to embrace type-safety, where do they put their units?  Health is a "facade/factory" for these domain specific units like BPM, Steps or Floors. As of now we have a QuantityFactory and RI also contains a convenience façade Quantities. So far SystemOfUnits implementations aside from parsing are the entry points to access a desired unit.  What do you suggest instead? JavaDoc also states: "Nothing prevents a unit from belonging to several systems of units at the same time (for example an Imperial system would have many of the units held by the US Customary system)." so Health or other vendor-specific unit systems are not meant to replace SI and if they contained something like "Beats per Second" or "Beats per Minute" then you may find SECOND in both of them just like the JavaDoc says. If we want to enforce SystemOfUnits only for "government accepted" global unit systems like SI, US and a few others, we must make that clear, but it could pervert the idea of having such interface there in the first place. Defining 2 or 3 final classes in the RI (or elsewhere if we manage to abstract it from the RI well enough to put it into API) would be enough, or see Fitbit (or uom-impl-enum) we may just consider using an enum after all |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Dec/14 ] |
| Leaving the question of existence for SystemOfUnits aside, let's stick to the Health example, so SECOND, BEAT, FLOOR and STEP would be likely Base Units in that system, while BPM, "Steps per Hour" or "Floors per Day" wouldn't since they are all derived from the others?  Right? |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 17/Dec/14 ] |
| Right, "Steps per hour" is a unit derived from two base units: step and second. The getBaseUnits() method would return STEP and SECOND (among others) while getUnits() would return a larger set.  Repeating myself: the set returned by getBaseUnits() is nothing else than the set of all possible keys in the map returned by Unit.getProductUnits(). The "product units" map does not contains arbitrary units - the javadoc explicitly said that it contains base units (otherwise "product units" would be less useful). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Dec/14 ] |
| Ok, that helps and a somewhat extensive explanation would also be necessary in JavaDoc.  Can you explain where this method would help archive something that the other method doesn't do? E.g. any base unit instance in SI is a BaseUnit, so instanceof BaseUnit works to find out its type. Where would this collection help developers more?  Especially for mainly derived systems like US it contains public static final Unit<Length> METER = METRE, but all other base units are used from SI like KILOGRAM and all relevant US units in the system like POUND are derived units, so this method would contain mostly the same units SI would expose there (except METRE called METER in this system ) Even if SI could ever move into the API any unit system could then potentially have interdependencies with other systems defined maybe not even in the same package or module. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Jan/16 ] |
| Using the appropriate profile, <https://github.com/unitsofmeasurement/unit-tck/blob/master/src/main/java/tec/units/tck/tests/quantity/QuantityTypesTest.java> (testContainsBaseDimensions() tests if all SI base units / dimensions are available in a set of units provided by an implementation.  Beyond testing this in the TCK, I am not sure, if such method really adds value to the API. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 23/Feb/16 ] |
| I understand what the method should return, but I also don't see a use case where this would be used. Martin, do you have a use or test case to share where the added method would be useful?  Regards, Leo. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Jun/16 ] |
| We do have the same method on unit now. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-87] [Merge the model package into quantity and root](https://java.net/jira/browse/UNITSOFMEASUREMENT-87) Created: 15/Dec/14  Updated: 24/Dec/14  Resolved: 20/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16886) |
| **Fix Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17107), [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, refactoring |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The "model" package is currently of lesser importance. Except implementing the Dimension interface it contains a helper type DimensionalModel which is used in one or two places internally by the RI but does not implement any part of the API.  Thus the plan is, to move the abstract base class DimensionalModel into the root package "ri" similar to all other abstract base classes (and one or the other interface) for consistency. The remaining two classes seem best fit into "quantity" also because QuantityDimension also refers to that name. |

|  |  |
| --- | --- |
| **[Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)** ([UNITSOFMEASUREMENT-3](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)) [UNITSOFMEASUREMENT-86] [Getting rid of java.util.Locale and ResourceBundle](https://java.net/jira/browse/UNITSOFMEASUREMENT-86) Created: 15/Dec/14  Updated: 11/Jun/15  Resolved: 11/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16886) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Attachments:** | How to use ResourceBundle for localization in Java ME - Wiki.mht |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-7~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-7) | Getting rid of java.lang.reflect | Resolved | | is related to | [~~UNITSOFMEASUREMENT-116~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-116) | Refactor UCUMFormat into a locale neu... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-103~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-103) | Look into Unicode CLDR | Resolved | |  |
| **Tags:** | embedded, format, i18n/l10n |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| While the API has already been relieved of remaining references to e.g. Locale there are a few more traces in the RI, especially related to formatting.  A key class used by formatting: <https://github.com/unitsofmeasurement/unit-ri/blob/master/src/main/java/tec/units/ri/format/SymbolMap.java> actually uses the Enumeration<String> of ResourceBundle.getKeys() for all relevant information, so if we pulled that argument out of the internals and pass an Enumeration (alternative could be 2 other methods returning a Set) instead of a ResourceBundle the RI would be free from class references not in ME 8.  Attached is a Nokia/MS page for localization in Java ME (IE format, sorry, but that's the easiest to get a single file, and Microsoft seems eager to get rid of former Nokia J2ME content, so I wanted to save the code snippets should we need them ) This way or similar the information could be retrieved and passed as a Set/Enumeration in a platform neutral way. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 16/Dec/14 ] |
| [UNITSOFMEASUREMENT-7](https://java.net/jira/browse/UNITSOFMEASUREMENT-7) has code from Java ME 8 SDK on this kind of thing... |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 16/Dec/14 ] |
| Contains a similar code example from Java ME 8 SDK |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-85] [Remove <N extends Number,U extends Unit<Q>> in QuantityFactory](https://java.net/jira/browse/UNITSOFMEASUREMENT-85) Created: 13/Dec/14  Updated: 23/Dec/14  Resolved: 15/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16886) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, method |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| In the current QuantityFactory interface, for the following method: <N extends Number,U extends Unit<Q>> Quantity<Q> create(N number, U unit) The <N extends Number,U extends Unit<Q>> parameterization is useless, as neither the N or U parameter is used in the return type or for parameterization of other parameters. I suggest to simplify the method signature as below (not also the renaming of 'number' as 'value' for consistency with Quantity property names): Quantity<Q> create(Number value, Unit<Q> unit) |

|  |  |
| --- | --- |
| **[Make RI and related libraries available to public Maven repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-35)** ([UNITSOFMEASUREMENT-35](https://java.net/jira/browse/UNITSOFMEASUREMENT-35)) [UNITSOFMEASUREMENT-84] [Make RI available on MavenCentral](https://java.net/jira/browse/UNITSOFMEASUREMENT-84) Created: 11/Dec/14  Updated: 04/Mar/15  Resolved: 04/Mar/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Cloners** | | | | | is cloned by | [~~UNITSOFMEASUREMENT-123~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-123) | Make libraries available on MavenCentral | Closed | |  |
| **Tags:** | deployment, maven |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **[Make RI and related libraries available to public Maven repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-35)** ([UNITSOFMEASUREMENT-35](https://java.net/jira/browse/UNITSOFMEASUREMENT-35)) [UNITSOFMEASUREMENT-83] [Make RI and libraries available on Bintray.com](https://java.net/jira/browse/UNITSOFMEASUREMENT-83) Created: 11/Dec/14  Updated: 13/Dec/14  Resolved: 13/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | deployment, maven |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-82] [EDR](https://java.net/jira/browse/UNITSOFMEASUREMENT-82) Created: 10/Dec/14  Updated: 28/Jan/15  Due: 27/Jan/15  Resolved: 28/Jan/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | review |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Early Draft related activities |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 28/Jan/15 ] |
| EDR finished |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-81] [Fix LocalUnitFormat.formatInternal()](https://java.net/jira/browse/UNITSOFMEASUREMENT-81) Created: 01/Dec/14  Updated: 01/Dec/14  Resolved: 01/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | conversion, format, i18n |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Apply <https://github.com/unitsofmeasurement/uom-se/issues/49> from the SE port accordingly. This internal method should not contain anything SE 8 specific, so most of it is expected to work on ME, too. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-80] [javax.measure.spi.UnitFormatService uses java.util.Locale](https://java.net/jira/browse/UNITSOFMEASUREMENT-80) Created: 29/Nov/14  Updated: 01/Dec/14  Resolved: 01/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |
| **Environment:** | Java ME 8.1 GA | | |

|  |  |
| --- | --- |
| **Tags:** | embedded, format, i18n |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| javax.measure.spi.UnitFormatService uses java.util.Locale, which is not available |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Dec/14 ] |
| Will remove it, thanks. The idea was to retrieve specialized versions of UnitFormat like LocalUnitFormat via Locale but we'll have to consider alternatives, e.g. pass locale strings as optional parameters if necessary. At the moment we'll remove it and see, whether it can be handled implementation-specific in this case or additional ME-compliant arguments are necessary. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Dec/14 ] |
| When fixing this, JavaDoc should also be adjusted. While UCUM is one option for a format it is not a good default. There are alone 3 variants and people normally expect something human readable like "10 kg" rather than a UCUM code. Thus "see UCUM Full Specification" should be removed from JavaDoc. Same for the getUnitFormat() signature where "Returns the default UCUM unit format." shall be replaced by something more neutral. getUnitFormat("UCUM") from the same signature could be used as an example in the getUnitFormat(String name) signature, but should not be considered default. That's up to each implementation. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-79] [Add contains() method to Range](https://java.net/jira/browse/UNITSOFMEASUREMENT-79) Created: 26/Nov/14  Updated: 19/Jan/15  Resolved: 17/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16886) |
| **Fix Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17107), [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | operations, value, values |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The Range class of the RI is modeled somewhat after OpenXC, Google Guava or the old JSR 256 MeasurementRange.  Especially the Guava equivalent while more complex offers a (abstract?) method contains() taking a value and checking whether it is within that range. It could be helpful to take ComparableQuantity into consideration, possibly even restrict Range to use Quantity <Q> rather than fully generic arguments (like Guava or OpenXC do). |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-78] [Can we improve Formatting by adding some JSR 275/JScience methods](https://java.net/jira/browse/UNITSOFMEASUREMENT-78) Created: 25/Nov/14  Updated: 20/Sep/15  Resolved: 20/Sep/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-161~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-161) | Do we also want UnitFormat.alias()? | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-139~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-139) | Special GRAM format problem with sub-... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-143~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-143) | Multiples of HERTZ wrongly formatted | Resolved | | is related to | [~~UNITSOFMEASUREMENT-149~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-149) | Create UnitFormat.format() method wit... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-154~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-154) | How to tell, if UnitFormat is locale-... | Resolved | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-124~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-124) | Introduce a Unit "DB" | Resolved | |  |
| **Tags:** | format, i18n, question |
| **Epic Link:** | [View](https://java.net/jira/browse/UNITSOFMEASUREMENT-110) |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Formatting is a weakness at the moment. Not only because of Java's own type problems (no reification of generic types) but in some cases the result of a unit calculation is a new or currently unknown type that nevertheless deserves proper output and formatting.  JSR 275 offered at least 2 methods in UnitFormat<http://jscience.org/api/javax/measure/unit/UnitFormat.html> alias() and label(), which sounded a bit similar, see use cases for the label() method:  UnitFormat.getInstance().label(DAY.multiply(365), "year");  UnitFormat.getInstance().label(METER.multiply(0.3048), "ft");  You get the idea. If either for Unit directly or Quantity we found a similar way to address non trivial formatting problems like the SmartHomeDemo brought up, it seems like a great improvement. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Mar/15 ] |
| Methods like alias() or label() also existed in JSR 108 already. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Jul/15 ] |
| At least under Java SE 8 (1.8.0, drone.io never upgraded to later builds) on drone.io CI server, the label() method call showed a strange behavior:  [ERROR] COMPILATION ERROR :  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1092,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1093,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1094,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1095,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1096,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1097,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1098,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1099,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  (argument mismatch; inference variable Q has incompatible bounds  equality constraints: javax.measure.quantity.Volume  upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.3:compile (default-compile) on project uom-se: Compilation failure: Compilation failure:  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1092,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1093,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1094,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1095,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1096,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1097,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1098,22] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] /home/ubuntu/src/github.com/unitsofmeasurement/uom-se/src/main/java/tec/uom/se/format/SimpleUnitFormat.java:[1099,24] no suitable method found for label(javax.measure.Unit<javax.measure.quantity.Volume>,java.lang.String)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#1 of ?>,javax.measure.Quantity<Q>)  [ERROR] method tec.uom.se.format.SimpleUnitFormat.DefaultFormat.label(javax.measure.Unit<?>,java.lang.String) is not applicable  [ERROR] (argument mismatch; inference variable Q has incompatible bounds  [ERROR] equality constraints: javax.measure.quantity.Volume  [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q>)  [ERROR] -> [Help 1]  [ERROR]  [ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.  [ERROR] Re-run Maven using the -X switch to enable full debug logging.  [ERROR]  [ERROR] For more information about the errors and possible solutions, please read the following articles:  [ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException  Even more strange, because the method signature of format() uses the same Unit<?> wildcard, which here is also necessary, there can be any unit and also results of arithmetic operations to be formatted, labelled or aliased.  Will keep an eye on it, but it's not even clear, if this persists on more recent versions of Java SE 8 or SE 9 previews. Never happens in SE 7 either. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Jul/15 ] |
| Some of these links seem related to the issue: <http://stackoverflow.com/questions/23063474/why-does-this-java-8-program-not-compile> or <http://stackoverflow.com/questions/30622759/inference-variable-has-incompatible-bounds-java-8-compiler-regression> and JIRA tickets like <https://bugs.openjdk.java.net/browse/JDK-8043926> As it was said to be "Fixed" under OpenJDK, there's evidence, most recent service packs of SE 8 should behave correctly (e.g. if I compile the same SimpleUnitFormat under uom-se locally with JDK 1.8.0\_45 I get no errors) |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-77] [Could QuantityFactory safely return sub-types of Quantity rather than Quantity<Q>?](https://java.net/jira/browse/UNITSOFMEASUREMENT-77) Created: 20/Nov/14  Updated: 22/Jul/16  Resolved: 22/Jul/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Trivial |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-66~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-66) | Add asType() to Quantity | Resolved | |  |
| **Tags:** | design, question, types |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 22/Jul/16 ] |
| Guess we have to defer that to another version. Possibly not until Java is ready via Project Valhalla or similar. Could have an impact on a future version of this JSR but for now it seems unlikely without compatibility and conversion issues. |

|  |  |
| --- | --- |
| **[Make API available to public Maven repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-34)** ([UNITSOFMEASUREMENT-34](https://java.net/jira/browse/UNITSOFMEASUREMENT-34)) [UNITSOFMEASUREMENT-76] [Make API available on MavenCentral](https://java.net/jira/browse/UNITSOFMEASUREMENT-76) Created: 13/Nov/14  Updated: 29/Dec/14  Resolved: 29/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Cloners** | | | | | clones | [~~UNITSOFMEASUREMENT-75~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-75) | Make API available on Bintray.com | Resolved | |  |
| **Tags:** | Maven, deployment, repository |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Dec/14 ] |
| EDR made available, see <http://search.maven.org/#search%7Cga%7C1%7Ca%3A%22unit-api%22> |

|  |  |
| --- | --- |
| **[Make API available to public Maven repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-34)** ([UNITSOFMEASUREMENT-34](https://java.net/jira/browse/UNITSOFMEASUREMENT-34)) [UNITSOFMEASUREMENT-75] [Make API available on Bintray.com](https://java.net/jira/browse/UNITSOFMEASUREMENT-75) Created: 13/Nov/14  Updated: 25/Dec/14  Resolved: 25/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886), [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17107) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-74~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-74) | Use Bintray.com | Resolved | | **Cloners** | | | | | is cloned by | [~~UNITSOFMEASUREMENT-76~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-76) | Make API available on MavenCentral | Resolved | |  |
| **Tags:** | Maven, deployment, repository |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-74] [Use Bintray.com](https://java.net/jira/browse/UNITSOFMEASUREMENT-74) Created: 13/Nov/14  Updated: 21/Nov/14  Resolved: 21/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-75~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-75) | Make API available on Bintray.com | Resolved | |  |
| **Tags:** | Maven, deployment, repository |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| A JavaMoney contributor pointed us to Bintray.com for snapshot and other deployments. It seems backed or run by Artifactory and its maker, so probably an alternative to MavenCentral with a business model similar to GitHub. While for some experimental stuff an "embedded" repository here seems nothing wrong, even for snapshot builds we should try to publish all relevant artifacts there, too. Manually and as a next step via CI server. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Nov/14 ] |
| Created a user "uom" and a related organization "unitsofmeasurement" which holds the key repository: <https://bintray.com/unitsofmeasurement/maven> (though for certain packages, uom might also use its own repositories created there by default ) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 21/Nov/14 ] |
| Bintray user and organization for JSR 363 created. |

|  |  |
| --- | --- |
| **[Rethink hierarchy (and possibly naming) of Quantity and Measurement](https://java.net/jira/browse/UNITSOFMEASUREMENT-69)** ([UNITSOFMEASUREMENT-69](https://java.net/jira/browse/UNITSOFMEASUREMENT-69)) [UNITSOFMEASUREMENT-73] [Refactor Quantity](https://java.net/jira/browse/UNITSOFMEASUREMENT-73) Created: 02/Nov/14  Updated: 05/Nov/14  Resolved: 05/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, refactoring |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| So far the intended goal is, to put Quantity on top of the hierarchy and merge the Measurement type into it.  (option 1. proposed by Jean-Marie in the mailing list) |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-72] [Move some elements in "format" to RI/other impl](https://java.net/jira/browse/UNITSOFMEASUREMENT-72) Created: 02/Nov/14  Updated: 02/Nov/14  Resolved: 02/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, format |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Similar to JavaMoney, some enums in the "format" bundle like FormatBehavior look like they serve the implementation only, so we 'd move them to RI for now. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-71] [Should we rename of(CharacterSequence) to parse()?](https://java.net/jira/browse/UNITSOFMEASUREMENT-71) Created: 31/Oct/14  Updated: 15/Nov/14  Resolved: 15/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-67~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-67) | Convenience Factory Methods for Quant... | Resolved | |  |
| **Tags:** | factory, parsing |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This affects only the RI or similar implementations. JSRs like 354 (or 310) started calling a convenience factory method that takes a CharSequence (String)  public static Money parse(CharSequence text) {  ...  }  As of now similar methods in JSR 363 implementations are called of(), but in many cases those methods in AbstractUnit or similar implementing classes call formatters and their parse() methods under the hood, so calling those methods parse(), too could be a logical alternative. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Nov/14 ] |
| Martin Fowler also suggests to call it parse() in his theorem on Quantity <http://martinfowler.com/eaaDev/quantity.html> |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 08/Nov/14 ] |
| To CharSequence much methods have using this nomenclature. Also I like the parse method. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-70] [Add five methods in Quantity interface](https://java.net/jira/browse/UNITSOFMEASUREMENT-70) Created: 31/Oct/14  Updated: 03/Nov/14  Resolved: 02/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) | **Assignee:** | [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | arithmetic, design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The propose is add five new comparatives methods in Quantity interface:  boolean isGreaterThan(Quantity<T> quantity);  boolean isGreaterThanOrEqualTo(Quantity<T> quantity);  boolean isLessThan(Quantity<T> quantity);  boolean isLessThanOrEqualTo(Quantity<T> quantity);  boolean isEquivalentTo(Quantity<T> quantity);  The implementation should convert to the same unit and than compare. So:  Quantity<Time> oneHour = ...  Quantity<Time> sixtyMinutes = ...  **Should return true, when:**  oneHour.isEquivalentTo(sixtyMinutes);  In another JSRs such JSR 354 and JSR 310 there are comparatives methods:   * <https://github.com/JavaMoney/jsr354-api/blob/master/src/main/java/javax/money/MonetaryAmount.java> * <http://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html#isAfter-java.time.LocalTime-> * <http://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html#isBefore-java.time.LocalTime-> |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 31/Oct/14 ] |
| Looking at even the FastMoney implementations of these methods in Moneta like:  public boolean isLessThan(MonetaryAmount amount) {  checkAmountParameter(amount);  return getBigDecimal().compareTo(amount.getNumber().numberValue(BigDecimal.class)) < 0;  }  None of them seem to get to a result without BigDecimal. That looks like a waste for the general API level Quantity or Measurement depending on where operations end up.  Keep in mind, LocalDateTime in JSR 310 are IMPLEMENTATION classes of an RI boggled up with an anemic hard to distinguish API, but the few interfaces like Temporal, TemporalAmount or TemporalUnit can be seen as minimalistic API, and neither of these contain more than a few like plus() or minus() (a Scala style, that has not been accepted anywhere else, hence 354 or this JSR prefer the pattern by java.math classes ) So it feels, like keeping them e.g. in implementation classes like DecimalQuantity or no further than AbstractQuantity looks better. Happy to see what other EG members feel, but based on mostly RI locations in other mentioned JSRs I'm hesitant to think it adds enough value to the API. In 354 it is mostly for precision math with BigDecimal, something we don't have in ME. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 01/Nov/14 ] |
| Why not just implement the int compareTo(Quantity<T>)) method from the Comparable interface instead? |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 01/Nov/14 ] |
| About Comparable interface, we already had this discussion and the answer was not. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 02/Nov/14 ] |
| At the time we had that discussion, they were no proposal to add comparison methods in Quantity. Now this proposal change the situation, we should at least revisit the arguments against Comparable. I have heard so far:   * Quantity shall be implementable by Enum. Okay. But most Quantity methods like getValue(), getUnit(), add, multiply make no sense for enumeration like LIGHT or HEAVY. Is Quantity really the right interface for those enumerations?   Does someone remember other arguments?  If the decision still to not extend Comparable and to provide different comparison purpose, then I think that we should provide 5 methods. We should provide only 1, maybe something like compareTo but returning an Enum instead than an int. The Enum values could be LESS, EQUIVALENT (not exactly the same than "equals" in the Java sense), GREATER or NOT\_COMPARABLE. Having 5 separated methods double the API weight and is a performance problems, since sorting Quantity objects requires calling 2 or 3 methods for many pair of objects (e.g. isGreater, followed by isEquals or isLess if the former returned false. Invoking a third methods may be necessary if we want to support non-comparable values like NaN). |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 02/Nov/14 ] |
| It could not be useful to user, how just comparator that return a boolean. How add these methods it can impact in performance? Probably, the implementation will use some thing like compareTo.  To sort already exist the Comparator interface. In SE implementation, for example, we have the QuantityFunctions that has five kinds of sort, I believe it don't will problem put these comparators, doing the downgrade, in ME platform (one or all). |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 02/Nov/14 ] |
| @keilw , Sorry I don't understand your point, I am not talking about precision or not, just method to do comparatives. Like this:  Quantity<Time> oneHour = ...  Quantity<Time> sixtyMinutes = ...  if (oneHour.isEquivalentTo(sixtyMinutes)) {  ...  }  Quantity<Length> grams = ...  Quantity<Length> kiloGramas = ...  if (kiloGramas.isLessThan(grams)) {  ...  }  The implementation should convert and than compare the value. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Nov/14 ] |
| No, Quantity in its current form is a Magnitude (or the Quantity, OpenXC has) while the abstract base type Measurement (that's what OpenXC calls Unit ) can and should be implemented by Enums, too.  In theory Quantity could also extend Comparable, but see JSR 310 most of its concrete classes do, not the more or less API classes like TemporalAmount. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 02/Nov/14 ] |
| Otavio: about performance, I'm not saying that adding those methods would degrade performance of other methods. I'm saying that a bunch of methods like:   * boolean isGreaterThan(...) * boolean isLessThan(...) * etc.   is less performant in some scenarios than a single method like:   * int compareTo(...) (or any other methods returning an int or an Enum if we do not want to clash with Comparable).   Sorting is not the only example. Any algorithm who needs to know if A is less, equals or greater than B will be impacted in the same way. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 02/Nov/14 ] |
| hum, Understood, thank you to explain me, but the focus of this method is just a single useful comparative and get the code more expressive.  For example:  This code is more expressive:  if (kiloGramas.isLessThan(grams)) {  ...  }  Then this:  if (kiloGramas.compareTo(grams) < 0) {  ...  }  Just to follow anothers new APIs, that are using method as DSL language. In Enterprise App, we always need to do a comparatives like these. To sort functions I believe the best strategy is implements the Comparator interface.  Considering the boolean in JVM does not exist and will be an int, I believe it will not lost performance, maybe in the worse scenario a slight loss of performance. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Nov/14 ] |
| Don't want to shut this down or resolve with "won't fix" but until the Quantity design and scope is finalized this makes no sense.  If the current trend, that Quantity shall keep all of its methods and Measurement is practically submerged into it, hence drop API support for non-numeric custom types (which would often be modelled as Enum) is executed, there could be a viable alternative to extend Comparable<Quantity> and IMHO that would seem a bit leaner.  If implementations really need this, feel free to add it to AbstractQuantity in the SE port to see what value it adds. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 02/Nov/14 ] |
| Ok, I specializes to SE. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 02/Nov/14 ] |
| Just to SE |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-69] [Rethink hierarchy (and possibly naming) of Quantity and Measurement](https://java.net/jira/browse/UNITSOFMEASUREMENT-69) Created: 31/Oct/14  Updated: 04/Mar/15  Resolved: 20/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [dautelle](https://java.net/jira/secure/ViewProfile.jspa?name=dautelle) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-65~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-65) | Move QuantityFactory from "function" ... | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-73](https://java.net/jira/browse/UNITSOFMEASUREMENT-73) | [Refactor Quantity](https://java.net/jira/browse/UNITSOFMEASUREMENT-73) | Sub-task | Resolved | keilw | |  |
| **Tags:** | api, inheritance |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Jean-Marie pointed out what he feels is a hierarchical inversion Measurement should extend Quantity and not the vice versa (a measurement is a quantity "measured"; hence with a value stated in a unit)  and suggested  interface Measurement<Q> extends Quantity<Q> {  Number getValue()  Unit<Q> getUnit()  toUnit(Unit<Q>) // toUnit should be here and not in the Quantity interface (a quantity intrinsically has no unit)  }  Wikipedia: <http://en.wikipedia.org/wiki/Quantity> makes distinction between quantities with magnitute or multitude.  Furthermore "..., the measurements of quantities" goes alongside Jean-Marie's proposal to swap them. Eclipse UOMo also knows an (Eclipse Interface style) IMeasure <http://git.eclipse.org/c/uomo/org.eclipse.uomo.git/tree/bundles/org.eclipse.uomo.units/src/main/java/org/eclipse/uomo/units/IMeasure.java> extending Quantity of the earlier Unit-API.  If we want to cover non-numeric quantities and Measurement was therefore a numeric quantity would the Quantity interface therefore be nothing more than a "tagging Interface" for the type-safe generic element <Q>? If the numeric value and unit as well as to() (which is strictly related to a getUnit() method) all went from Quantity to a sub-Interface Measurement there is no method left.  Although it is arguable, if say a T-Shirt size like "M" or "XL" can often be simply converted into a numeric Clothes measurement like "45" or "52", such conversions do exist in real life. It often tends to be a Range which we only Support on an implementation level (in RI or alternate implementations like the SE port) so unless Range would implement at least Qantity or Measurement itself, a conversion may exceed the scope of the API. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 31/Oct/14 ] |
| Only for a reference, not to apply here there are some bizarre hierarchies, just to mention OpenXC by Ford: <http://android.openxcplatform.com/reference/com/openxc/units/Quantity.html> Where Quantity extends Unit which then has concrete subtypes named Liter or Kilometer. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 01/Nov/14 ] |
| I'm not aware of universally accepted definition of "quantity". Some peoples make a distinction between "quantity" and "physical quantity". For example in [an article cited by Werner](http://research.microsoft.com/en-us/um/people/akenn/units/RelationalParametricityAndUnitsOfMeasure.pdf), the author first mentions "physical quantity", then use only the word "quantity" in the remaining of his article. He said "Typically these quantities possess unit of measure", but I think that he implicitly means "physical quantity".  One could imagine the following type hierarchy:  Quantity  ↑  Measurement  ↑  PhysicalQuantity or Scalar or ScalarMeasure  But where would we put Vector, Tensor or any other kind of measurement that are not a scalar in this hierarchy? Putting Vector as a subtype of Measurement seems contradictory with the Quantity base type.  There is my proposal:   * I think that Measurement should not extend Quantity, because (to me) it would imply that all measurements are scalars, which would exclude other kind of measurements like vectors. * I think we should explain that, for the sake of simplicity, JSR-363 merges the "pure quantity" concept with the "physical quantity" concept, and that we call that just "quantity". We may define our quantity as a "quantitative measurement".   A Quantity in the "pure" sense is hard to fit in our type hierarchy (e.g. above-cited contradiction with non-scalar measurement) and make all Length, Mass, etc. subtypes useless for other purpose than marking interface. Furthermore the above hierarchy would allow users to parameterize Unit instances with arbitrary Measurement subtypes that are not something like Length or Mass. On the other hand, keeping the current hierarchy, with the idea that we merged "quantity" concept with "physical quantity" or "quantitative measurement", keep Unit parameterization safer (can not use arbitrary Measurement), and keep Measurement applicable to non-scalar measurement.  In summary, I suggest to keep current hierarchy as-is but explain better what we means by "quantity". |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Nov/14 ] |
| Was resolved by removing Measurement from the API. Current design of Quantity assumes we only support <b>numeric</b> quantities, not things like "Small, Medium, Large", etc. There are means to support this in Java, especially enums (we might want to mention that in the spec, too ), thus it can be seen as low or no priority for this JSR.  Unlike OSGi Measurenent, we consider Quantity and Measurement as 2 separate entities. Also see Martin Fowler's analysis: <http://martinfowler.com/apsupp/apchap3.pdf>  Measurement can contain Quantity rather than extending it. There's room for this in implementations, especially since methods like <b>timestamp</b> can be different on ME or SE. Should there be a need to graduate such type into the API we'll consider it either at a later stage of this JSR or a later version. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Mar/15 ] |
| Just for the record, a fork of the simple UCAR Unit library into another project named **Visad** also contains a Quantity type: <https://www.unidata.ucar.edu/software/idv/docs/javadoc/ucar/visad/quantities/Quantity.html> Actually there are 3 but this one has children ScalarQuantity, VectorQuantity similar to what was discussed here and ScalarQuantity got sub-classes like Acceleration, Length, etc., very much like we do here except that we currently don't distinguish between Scalar and Vector, effectively all the API supports is just ScalarQuantity for now. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-68] [Remove DimensionService](https://java.net/jira/browse/UNITSOFMEASUREMENT-68) Created: 31/Oct/14  Updated: 01/Nov/14  Resolved: 01/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As spoken on the Mailing list, DimensionService makes little sense, since Dimension is normally called from other elements like Unit. It seems a candidate for pruning. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 31/Oct/14 ] |
| Just to clarify possible side-effects. RI and similar implementations implement Dimension in a class QuantityDimension (the name reflects "The dimension of a physical quantity..." see <http://en.wikipedia.org/wiki/Dimensional_analysis#Definition>) That class exposes the Basic physical dimensions as constants. Especially SystemOfUnits has a getUnits(Dimension method where a Dimension argument is required. required argument are normally within those constants, there is really no need to provide them like DimensionService intended to.  Should custom dimensions be an important requirement, let's think where those would come from, e.g. simply new MyDimension() based on something like  class MyDimension implements Dimension {  ... |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Nov/14 ] |
| Removed |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-67] [Convenience Factory Methods for Quantity or Unit](https://java.net/jira/browse/UNITSOFMEASUREMENT-67) Created: 26/Oct/14  Updated: 15/Nov/14  Resolved: 15/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-65~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-65) | Move QuantityFactory from "function" ... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-71~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-71) | Should we rename of(CharacterSequence... | Resolved | |  |
| **Tags:** | api, design, factory |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Along the lines of a proposed Java 9 JEP (<https://bugs.openjdk.java.net/browse/JDK-8048330>) to add convenience methods to Java Collections types like List, Map, Set or their implementing classes like ArrayList.  As of now, the API must support both Java SE and ME (8), so adding a static method like of() to an interface is not an option (could be if Java ME 9 or 8.x should ever support Lambdas as Mark Reinhold hinted )  The closest to an interface are abstract base classes like AbstractUnit or AbstractQuantity in the RI.  AbstractUnit contains an of() convenience factory for strings like "10 cm". while for Quantity implementations it was moved to BaseQuantity. IMHO a mistake, because  public static <Q extends Quantity<Q>> AbstractQuantity<Q> of(double, Unit<Q>)  in BaseQuantity actually returns a new DoubleQuantity instance.  DoubleQuantity does not extend BaseQuantity but AbstractQuantity, thus the of(double, Unit<Q>) method should either be in the concrete DoubleQuantity class (which however is not public right now) or AbstractQuantity not BaseQuantity.  Doing so on AbstractQuantity and AbstractUnit seems more consequent towards a possible future addition to the actual interface (if Lambdas worked for ME, too) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 30/Oct/14 ] |
| If Quantities returned a Quantity instance directly, then both convenience factory methods in concrete or abstract types become less useful and it means, QuantityFactory would better be OPTIONAL via SPI. Should the facade return QuantityFactory like JSR 354 does in similar classes, then it remains a first class API type similar to JavaMoney MonetaryAmountFactory |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 01/Nov/14 ] |
| I propose:   * QuantityFactory as an interface doing the real work. Users are responsible for fetching their QuantityFactory instance using java.util.ServiceLoader or OSGi. * Quantities as a set of **static** methods only, which are only convenience methods fetching a default QuantityFactory using ServiceLoader (or whatever mechanism) and delegating the work to it. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Nov/14 ] |
| Thanks, that sounds reasonable. Otavio has recently through other tasks (mostly in the SE port) refactored BaseQuantity into NumberQuantity. As it reflects aspects of JavaFX Binding, it seems good for RI or similar implementations (leaving aside in our current design EVERY Quantity is also a numeric quantity since getValue() only returns a number now ) When this "base quantity" (the only concrete class that is fully exposed to sub-classing, although a user could directly use AbstractQuantity for more freedom if they want) is subclassed, overloading the static of() method becomes tricky and cumbersome, so I suggest to remove that for now. Other factories for specific types, maybe we could leave. With a new name like NumberQuantity while they are not subclasses it delegating to special (primitive) numbers like long or double sounds OK.  All other factory methods should happen in Quantities or a similar specialized static facade.  Java 9 won't remove Collections either, even if it may add new of() methods to the Interface itself. We won't risk more reduncancy than that and in a hypothetical JDK module in the future, maybe there's ME 9 offering Lambdas, so we could offer that kind of stuff to a 1.x or 2.x Version of the API instead |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-66] [Add asType() to Quantity](https://java.net/jira/browse/UNITSOFMEASUREMENT-66) Created: 20/Oct/14  Updated: 20/Nov/14  Resolved: 20/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-77~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-77) | Could QuantityFactory safely return s... | Resolved | |  |
| **Tags:** | arithmetic, quantity, types |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| To provide type checking and a safer alternative to a cast, Quantity should also have an asType() method. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 01/Nov/14 ] |
| I think that casting to an explicit type like (Length) or (Mass) would be cleaner. It still not clear to me why the implementation can not create instance of the right type. It is because the Unit implementation has no Class<Q> information associated with it? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Nov/14 ] |
| Signature  <T extends Quantity<T>> T asType(Class<T> type);  proposed instead |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Nov/14 ] |
| Because by casting you can do any "crap", see a patch of <https://github.com/unitsofmeasurement/uom-demos/blob/master/domain/health/src/main/java/tec/uom/demo/health/BMIDemo.java>  //Quantity<Volume> squareHeight = height.multiply(height).asType(Volume.class);  Quantity<Volume> squareHeight = (Quantity<Volume>) height.multiply(height);  The first line won't run without a runtime exception, the second one will, though the result is wrong. So asType() has its value, though it may not be usable in every situation, there could be a need for an explicit cast, but for "standard situations" mainly what OSGi Measurement would also cover, we have a more typesafe approach even at runtime. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Nov/14 ] |
| Trying to cast to Volume:  Volume squareHeight = (Volume) height.multiply(height);  will fail for another reason:  Exception in thread "main" java.lang.ClassCastException: tec.units.ri.quantity.DoubleQuantity cannot be cast to javax.measure.quantity.Volume  at tec.uom.demo.health.BMIDemo.main(BMIDemo.java:42)  And IMHO the type of the general purpose operations is likely doomed to fail a typesafe cast forever,  You could (since they are all derived from Quantity) try to pass in Length or Mass but the operation cannot instanciate any mutual concrete class except performing "Proxy Voodoo" and that is not possible on ME.  Hence, PoC is necessary for a substantial change to QuantityFactory in <https://github.com/unitsofmeasurement/unit-api/tree/master/src/test/java/javax/measure/test> first. Without Dynamic Proxies, etc.  Jean-Marie's proposal for something like Area = Length.multiply(Length) would work, but require explicit implementation of all these methods in a concrete class, see <https://github.com/unitsofmeasurement/unit-api/blob/master/src/test/java/javax/measure/test/quantity/DistanceQuantity.java> |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 03/Nov/14 ] |
| About the this method, I just would like to change the exception ClassCastException to anything more expressive maybe: InvalidUnitConvertionException  I prefer this strategy than specific methods, byond more implementations, we will limited the another implementations to just the API and Have the multiply and Quantity and in specific interface, IMHO no make sense. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Nov/14 ] |
| Well we should consider this carefully. The exception is thrown by classes like AbstractUnit so it is within our control, but similar to JSR 354 we prefer to reuse standard Java exceptions, i.E. throw an ArithmeticException in other cases, not some UnitArithmeticException  There is a UnconvertibleException (also extending the RuntimeException MeasurementException) but its current definition is closely bound to actual UnitConverter and conversion. Again, let's see, what others say, we should try to keep the number of special exceptions at a decent level, not trying to get to a level of UCAR (based on JSR 108) with 24 exceptions, and that API doesn't even cover the combination of units and values like ours, OSGi Measurement or ICU4J :-| |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 03/Nov/14 ] |
| I believe IllegalStateException, is better than ClassCastException and already is on JDK: <http://docs.oracle.com/javase/7/docs/api/java/lang/IllegalStateException.html> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Nov/14 ] |
| IllegalState makes no sense here, at most I could imagine IllegalArgumentException which is used in one or two places already.  JSR 354 MonetaryAmount states, "Arithmetic operations should throw an ArithmeticException", but e.g. one of those "compareTo() convenience methods" you also suggested for 363 @throws MonetaryException if the amount's currency is not equals to the currency of this instance.  The general problem sounds similar, but let's discuss, if we leave CCE, consider IllegalArgumentException a better choice or used MeasurementException the equivalent to MonetaryException here? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Nov/14 ] |
| Also this is a minor detail, the real challenge lies in a solid proof (JUnit tests in API, a selected number of concrete quantities can be found here: <https://github.com/unitsofmeasurement/unit-api/tree/master/src/test/java/javax/measure/test>) that changing at least QuantityFactory to return interfaces like Length rather than Quantity<Length will work without runtime exceptions.  Mostly ClassCastException btw. thus, IMHO asType() does it right at the moment. Class.cast() also does the same, and since the underlying Unit.asType() performs a cast to a desired unit, if that fails for whatever reason, you also get a ClassCastException so it seems more consistent, thatn throwing 2 different exceptions |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Nov/14 ] |
| Actual issue was fixed. Returning a different element from QuantityFactory or passing sub-types of Quantity instead of the parent interface to arithmetic operations is a different question that deserves a different task or multiple tasks. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-65] [Move QuantityFactory from "function" to "spi"](https://java.net/jira/browse/UNITSOFMEASUREMENT-65) Created: 12/Oct/14  Updated: 13/Nov/14  Resolved: 04/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-69~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-69) | Rethink hierarchy (and possibly namin... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-67~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-67) | Convenience Factory Methods for Quant... | Resolved | |  |
| **Tags:** | design, spi |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| In recent implementations, a Quantities facade along the lines of JSR-354 (JavaMoney) returns new Quantity instances without the use of QuantityFactory. It still seems to have its place (please advice if you feel otherwise, it could also result in chosing one over the other) e.g. if you look at implementations we had that use reflection to dynamically invoke new types.  If you draw this further, using QuantityFactory one may even instanciate new types defined at runtime via OSGi, while the Quantities singleton currently does not do that (it may of course use a factory underneath)  Could we move QuantityFactory into the (optional) SPI package? This means, implementations have a choice what they do. Especially on ME the whole "service" concept is optional by itself, one of the main reasons not to make SPI mandatory either. On SE or OSGi containers, implementations are free to use it or go the "simple" way of a static facade a'la Quantities |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 28/Oct/14 ] |
| Another alternative would be to let Quantities expose QuantityFactory, not return a Quantity directly, see JSR 354: <https://github.com/JavaMoney/jsr354-api/blob/master/src/main/JEP/MoneyAndCurrency.md>  MonetaryAmountFactory<Money> fact = MonetaryAmounts.getAmountFactory(Money.class);  Money m = fact.setCurrency("USD").setNumber(200.50).create();  is with minor differences (e.g. we need to parameterize with a Quantity subtype <Q>) very similar to what we do. In theory if the JSR 354 "with" chaining is preferred and appreciated, we could also tweak QuantityFactory implementations accordingly. The major difference is, the concrete class should remain in RI or other implementations to be platform neutral, e.g. Java SE vs. ME or OSGi, just to Name a few. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-64] [Make JSR available to Eclipse Orbit](https://java.net/jira/browse/UNITSOFMEASUREMENT-64) Created: 09/Oct/14  Updated: 23/Aug/15  Due: 30/Sep/15  Resolved: 18/Aug/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-58~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-58) | Consider a license change to BSD for ... | Resolved | | **Relates** | | | | | relates to | [JAVAMONEY-144](https://java.net/jira/browse/JAVAMONEY-144) | Resolve License issue with Eclipse | Open | |  |
| **Tags:** | eclipse, external, library |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As Eclipse IoT projects like SmartHome signalled interest in JSR 363 it should be made available to Eclipse Orbit, by filing a CQ (see <http://www.eclipse.org/projects/dev_process/ip-process-in-cartoons.php>) and where appropriate upload it to a P2 repo at Eclipse.org |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Jan/15 ] |
| CQ filed: <https://dev.eclipse.org/ipzilla/show_bug.cgi?id=9154> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Feb/15 ] |
| Smarthome recently added its own CQ, should get things going at the pace possible with Eclipse legal |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Mar/15 ] |
| Eclipse PMC still working on it. Likely nothing happens before EclipseCon, but promising signs. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Aug/15 ] |
| Eclipse IP team doesn't understand or approve the Spec License. Hence the API of JSR 363, 354 and 107 can't be used by Eclipse.org projects. So be it, then they need to stick to Unit-API Legacy forever |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-63] [Fix return value of Quantity.inverse()](https://java.net/jira/browse/UNITSOFMEASUREMENT-63) Created: 06/Oct/14  Updated: 20/Oct/14  Resolved: 20/Oct/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-62~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-62) | Could we specialize Generics for oper... | Resolved | |  |
| **Tags:** | design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| As Martin mentioned, the inverse() method of Quantity should not return <Q> as the return type may change according to the same method in Unit. The operation shall inverse the unit and where applicable numeric value, not just the latter. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 10/Oct/14 ] |
| I'll do this as it's related to #62 |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| Just for the record, the issue was mentioned by myself (Martin). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Oct/14 ] |
| See description |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Oct/14 ] |
| An important question is furthermore, if Quantity.inverse() shall only invert the Unit part or also perform 1/this on the numeric value? |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 13/Oct/14 ] |
| Of course it shall also invert the numerical value. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 13/Oct/14 ] |
| So that (2 Second).invert() == (0.5 Hertz) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Oct/14 ] |
| For actual result, please check <https://github.com/unitsofmeasurement/unit-impl-se/blob/master/src/test/java/tec/uom/se/DecimalQuantityTest.java> or DoubleQuantityTest in same package.  The return value is not changed to <?> since this has to be coordinated across the entire API (after we found a consensus on how ) but operations are performed on both value and unit now as mentioned.  The extra 50 or so tests for SE are a welcome addition. We'll require more unit tests in both RI and SE. Beyond that is of course a TCK. Let's cross that bridge after EDR. JSR 354 created a good example on how to do that for a non-platform JSR. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-62] [Could we specialize Generics for operations involving them?](https://java.net/jira/browse/UNITSOFMEASUREMENT-62) Created: 02/Oct/14  Updated: 20/Oct/14  Resolved: 20/Oct/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Attachments:** | QuantityWildcard.patch |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-61~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-61) | Hide Generics to avoid <?> | Resolved | | is related to | [~~UNITSOFMEASUREMENT-63~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-63) | Fix return value of Quantity.inverse() | Resolved | |  |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| While dropping gnenerics completely seems unacceptable as it would destroy the type-safety that made this API succesful and special compared to others out there, we could look into specializing the wildcard a bit more. See attached patch. It is just a quick test, if feasable we'd have to look at least at Unit in addition to Quantity since most relevant operations like divide() or multiply() affect both of them together. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Oct/14 ] |
| 62 may help to narrow the types you pass or get, but it does not suggest dropping generics, that would not meet the requirements for this operations. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 03/Oct/14 ] |
| IMHO, it is not a good option to use, cold means bad desing and we need to avoid the '<?>' operator with Generics:  Quantity<?> timeToEvacuate = l.divide(metricSpeed);  **Or**  Quantity timeToEvacuate = l.divide(metricSpeed);  Maybe using this way  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> divide(Quantity<T> that);  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> multiply(Quantity<T> that);  <T extends Quantity<T>> Quantity<T> inverse();  This way, I can do this way:  Quantity<Speed> timeToEvacuate = length.divide(time, Speed.class);  Quantity<Frequence> frequence = time.inverse(Frequence.class);  We can verify and catch an exception if the operation cannot be released. This way we need to use cast in inverse. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Oct/14 ] |
| As Jean-Marie (I recall) mentioned, inverse() shall be generic, too.  Terrence and also Leonardo made it rather clear, that for ME devices memory is precious and a cost factor.  A person's BMI (e.g. provided by a scale) is still relatively simple, and assuming your device can handle the entire "quantity" bundle of the API, those types would be available: <http://en.wikipedia.org/wiki/Body_mass_index> However your ME-powered scale won't be forced to define an explicit class/interface BMI with the proper generic output.  Carbon footprint: <http://en.wikipedia.org/wiki/Carbon_footprint> is more complex. And in many cases you have multiple intermediary results you don't want to have define types for. Not even on SE, but especially not on ME where space for such classes is more costly.  Thus on an API level, the generic argument must remain. Potentially (that's the scope of this ticket ) narrowing down the wildcard to Quantity or Measurement, but not more.  Implementations, especially on SE shall explore what you are worried about. Then we can see in future releases, if any of that was to benefit ME, but at the moment we must not blow up the API. ME 8 does not strip or obfuscate anything (that was specified so ) thus a simple "Hello World" demo we got at the moment is nearly 200kb. Without Debug info maybe less, but there is a good reason behind splitting MEEP 8 into several JARs it seems. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 06/Oct/14 ] |
| I added this PoC for BMI: <https://github.com/unitsofmeasurement/uom-demos/blob/master/domain/health/src/main/java/tec/uom/demo/health/BMIDemo.java> In theory one could even chain the operations like  Quantity<?> bmi = mass.divide(height.multiply(height));  System.out.println(bmi);  The important point is, it works without forcing an implementing system to define all sorts of type classes where mixed operations could have hundreds or even thousands of different possible results in real life.  In fact, the signature of Quantity  Quantity<Q extends Quantity<Q>>  already restricts the possible values for <?> to a subtype of Quantity.  Hence declaring anything more in multiply() or divide() seems redundant.  The question of adding some knowledge of every output type to operations should be explored in: <https://github.com/unitsofmeasurement/unit-impl-se/issues>  As seen with JSR 308 its approach looks very cumbersome and incomplete. E.g. the current Checkers framework knows metric length, and a Celsius or Kelvin, but no Fahrenheit temperature. Almost every element must be annotated, including a @cm vs. @m. There seems to be a significant flaw with this approach, where simply annotating @Length or @Temperature sounds reasonable, but doing so for each individual instance of each type looks almost unmaintainable.  Otavio please have a look if you feel it could work for your purpose |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 07/Oct/14 ] |
| Werner the proposal is just to avoid an unknown element, it's doesn't impact in performance, the implementation just will do a cast to you. Because the real proposal of generics is avoid you do cast. So before generics:  List integers = new ArrayList();  integers.add(12);  ...  Integer number = (Integer) integer.get(0);  After:  List integers = new ArrayList();  integers.add(12);  ...  Integer number = integers.get(0);  Using the same example that you did:  Quantity<Length> height = Quantities.getQuantity(1.87d, METRE);  Quantity<Mass> mass = Quantities.getQuantity(85d, KILOGRAM);  Quantity<Area> squareHeight = (Quantity<Area>)height.multiply(height);  Using generics:  Quantity<Length> height = Quantities.getQuantity(1.87d, METRE);  Quantity<Mass> mass = Quantities.getQuantity(85d, KILOGRAM);  Quantity<Area> squareHeight = height.multiply(height, Area.class);  The implementation just do cast to you. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Oct/14 ] |
| Area might be trivial as it still happens to be in the (yet optional on ME ) bundle of SI quantities, but the hundreds or thousands of new unknown types like "BMI" make the class argument useless while the generic one just works and gets users the desired outcome, e.g. their BMI.  Please explore in SE port for now, as it may just be an overhead in the ME context |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Oct/14 ] |
| Keep in mind, the "quantity" package alone is the biggest chunk of the API, with around 20kb of a total ~30kb API. Which is why it is optional. And therefore ME users should be able to cope with a minimum overhead. E.g. even define one or two types they need. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 07/Oct/14 ] |
| Werner. The javax.measure.quantity.Area there is in API. We don't need add nothing in SI.  The implementation just added a cast code, same thing like this:  **Before:**  @Override  public Quantity<?> divide(Quantity<?> that) {  return new DecimalQuantity(value.divide(toBigDecimal(that.getValue()),  MathContext.DECIMAL128), getUnit().divide(that.getUnit()));  }  **After:**  @Override  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> divide(Quantity<T> that, Class<E> unitType) {  return (Quantity<E>) new DecimalQuantity(value.divide(toBigDecimal(that.getValue()),  MathContext.DECIMAL128), getUnit().divide(that.getUnit()));  } |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Oct/14 ] |
| It is OPTIONAL, meaning we probably won't even have it in a final JAR or at least a compliant ME implementation can safely ignore it. The BMI example outlines clearly, that the end result is what people are interested in. Having a concrete type for the "throwaway" area value is not what the BMI calculation is about.  One can equally extend a BaseQuantity or AbstractQuantity from the SE port with any operation they want. Try adding these methods there, with or without JSR 308 or other means. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Oct/14 ] |
| How's an actual PoC in the SE port going? Large downstream users like DT and also prior contributors to JSR 275 or the UCUM Foundation all agree, a flexible handling of "Custom Units" is required. Hence, please demonstrate how an optional extra method (with extra arguments like class) may work in SE. It's perfectly fine to do that on AbstractQuantity level, there are other SE or RI specific methods there, too. You may need to handle it as Abstract\* e.g. add another method to Quantities or cast (just to get the Abstract class ) If this is succesful, we can judge if we add them to API. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 10/Oct/14 ] |
| Werner,  I, as a USER of the API, would like to write code like this:  Power instantPower = null;  Time integralizationPeriod = null;  Energy consumedEnergy = instantPower.multiply(integralizationPeriod);  Your example of BMI with Quantity<?> doesn't look much better than "float", as I would have no compile-time type safety, which is something I think we should aim. We would still have runtime safeties, I believe.  Otavio made a suggestion that we change the signature to be something like:  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> multiply(Quantity<T> that);  This doesn't enable the code I'd like to use, but enables the following, with is better than what we have right now:  Quantity<Power> instantPower = null;  Quantity<Time> integralizationPeriod = null;  Quantity<Energy> consumedEnergy = instantPower.multiply(integralizationPeriod);  I understand that this MIGHT break with new/unknwon units; we didn't explore that. But, still, I believe a provider of the API that would provide the new Interfaces. Is this a fair assumption? I think so.  So, I'd like Werner and the others opinions about this suggestion: remove the Quantity<?> multiply, divide and inverse, and change for a more specialized Generic form.  Just for completeness: To enable what I wanted in the beginning, we'd have to have specialized operations for common operations on each unit (for example, have the multiplication of power and time that results in energy, that's something we do; power and distance, no). That would increase the work and size of the API, and that may not be what we end up doing. But I also don't like the prospect of having a "BetterJSR363" that has this convenience methods... But this is something for another Jira issue |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Oct/14 ] |
| Leo,  I'm afraid the example you mentioned will not work with the general purpose QuantityFactory or the Quantities singleton so far introduced by Otavio, myself and (very ) few others, because it requires:  Quantity<Length> metre = Quantities.getQuantity(BigDecimal.TEN, SI.METRE);  not  Length metre = Quantities.getQuantity(BigDecimal.TEN, SI.METRE);  The latter won't work.  For concrete subclasses of BaseQuantity or enum based implementations like in <https://github.com/unitsofmeasurement/unit-impl-enum> this would work, because e.g.  public final class HeartRateAmount extends BaseQuantity<HeartRate> implements HeartRate {  There you could do  HeartRate hr = HeartRateAmount,of(10, BPM);  Of course, the concrete classes give you all freedom for custom operations one can hardly archieve by passing a class as suggested. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Oct/14 ] |
| I'll have a look at this suggested signature:  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> multiply(Quantity<T> that);  It might work, if all code compiles I'd commit it. If a use case still requires to pass <?> for <T> that sounds possible, so it would allow both flexibility of an unknown result where needed and a concrete type where available |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Oct/14 ] |
| Seems to work, at least RI and SE impl compile properly after the straw-men test with multiply().  Except classes that directly extend AbstactQuantity all code still seems to work. Meaning both  Quantity<Area> squareHeight = height.multiply(height);  or the previous version of BMIDemo  Quantity<?> squareHeight = height.multiply(height);  work equally fine. That seems least intrusive, especially compared to a version that enforces an extra class argument in the multiply() method. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 10/Oct/14 ] |
| Yes, less intrusive.  I started implementing it too. Are you going to implement this change in both API and RI, Werner?? So we don't work at the same thing at the same time  We need to change mostly everything with ? in it hehe.   * Quantity<?> for <E extends Quantity<E>> Quantity<E> METHOD () * Unit<?> for <E extends Quantity<E>> Unit<E> METHOD () * <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> METHOD (Quantity<T> tQuantity) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Oct/14 ] |
| Yes already done for multiply() but you're welcome to do divide() if you want.  I have to do a more significant change of all license headers of the SE port some time and would appreciate if it's possible to rename the repository from <https://github.com/unitsofmeasurement/unit-impl-se> to <https://github.com/unitsofmeasurement/uom-se> after that.  It would require everybody who checked it out (probably just me and you now ) to cut the connection and check it out again under the new name. All pushes should happen before that.  I will also create a ticket as either that type change or something else breaks 4 Unit tests in SE. Would be nice to look into that, too (but in Github where it belongs) |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 10/Oct/14 ] |
| Ok, I'll continue the work on my side. Will change API and RI.  About the SE Impl, that's Otavio's turf. I guess it's OK to do that now. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Oct/14 ] |
| Try it in Unit if you want.  Note, I used the signature  <T extends Quantity<T>, R extends Quantity<R>> Quantity<R> multiply(Quantity<T> that);  as <E> did not seem to make much sense (unless you speak German ) while <T> stands for either that or "type" while <R> represents "result".  In the asType() method of Unit we already use <T> but in theory one local method scope generic element should not erase or disturb one from another method. Please take special care of Quantity subtypes, some had a <T> earlier instead of <Q,> a few even with wrong JavaDoc pointing to <Q>. I sorted it all out, so now <Q> is the class level Quantity suptype, while <T> and <R> are used in methods like multiply() It does not keep you from doing something like  Quantity<Mass> squareHeight = height.multiply(height);  instead of <Area> btw., but as mentioned, a registry of all possible input and outcome types is probably beyond what we want to archieve. Even JSR 308 claims it cannot cover all possible types and every new individual operation like @kg or @lb (that does not exist in 308 Checkers yet ) would have to be explicitely declared.  Maybe if 308 ever worked for non primitive types we could explore that extra "safety harness" in the SE port. If it only works for primitives, we'd have to wait for "Valhalla" |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 10/Oct/14 ] |
| Ok, I'll do your signature. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| Sorry, I just saw this JIRA task. Please revert those changes. They **decrease** type safety. Examples:  **Previous method signature**  interface Quantity<Q extends Quantity<Q>> {  Quantity<?> multiply(Quantity<?> multiplier);  }  class Test {  void test() {  Mass mass = ...;  Length length = ...;  Quantity<?> r = mass.multiply(length); // Okay.  Quantity<?> q1 = ...;  Quantity<?> q2 = ...;  Quantity<?> q3 = q1.multiply(q2); // Okay.  }  }  **New method signature**  interface Quantity<Q extends Quantity<Q>> {  <T extends Quantity<T>, R extends Quantity<R>> Quantity<R> multiply(Quantity<T> that);  }  class Test {  void test() {  Mass mass = ...;  Length length = ...;  Quantity<Time> r = mass.multiply(length); // !!! Allowed by the compiler, but wrong !!!!  Quantity<?> q1 = ...;  Quantity<?> q2 = ...;  Quantity<?> q3 = q1.multiply(q2); // Not allowed anymore by the compiler !!!!  }  }  Same for the proposed change in Unit API: the compiler would happily accept second = kilogram.pow(2) for instance without any warning! This change does not improve type-safety, it completely destroy it!  I understand the wish to avoid <?> in generic method signatures, but there is simply nothing we can do in the Java language for any operation that do not return the exact same type, or a sub-type or a super-type of a method argument. This is not a bad design - this is a limitation of the Java language. The correct way to cast with JSR-363 is as in the following example:  Quantity<Speed> = length.divide(time).asType(Speed.class);  Note that this is the same idea than java.lang.Class.asSublass(Class) method. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 12/Oct/14 ] |
| One information, with new method code the doesn't broke.  Quantity<?> q1 = ...;  Quantity<?> q2 = ...;  Quantity<?> q3 = q1.multiply(q2);  Considering this method:  interface Quantity<Q extends Quantity<Q>> { Quantity<?> multiply(Quantity<?> multiplier);  }  The asType method would go to Quantity, so I could do same thing like this:  Speed speed = time.asType(Speed.class);  That really doesn't make sense. Because the variation to another quantity, should happen just in divide, multiply and inverse methods. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 12/Oct/14 ] |
| Otavio is right, demos like BMI or Hurricane category work fine in both. Previously to enforce a wrong result like  Quantity<Mass> timeToEvacuate = distanceKm.divide(speed);  you just had to cast. <?> as result still works, unlike the <Q> from the Quantity class signature, the <T> or <R> in operations can be substituted with <?> where required. We can't enforce the return type related to operators, at least not without adding "Voodoo" like JSR 308 type annotations if it'll ever work with reference types? Otherwise we'd have to wait for Valhalla, then a Java 9+ Quantity value type could be equal to primitives. A main reason for the license change. I spoke to Brian Goetz about it at the JCP party. His keynote 2 hinted it could make it to Java 9, at the party he sounded more like 10, but maybe there's either 9.x update or some things can be done faster. Anyway the SE port allows to explore that at any time. E.g. we could branch it for OpenJDK 9. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| With the method signature shown in the "New method signature" block of my above comment (which is the signature in the source code repository at the time of writing), attempt to compile the q1.multiply(q2) example on JDK8 produces the following error message:  incompatible types: inferred type does not conform to upper bound(s)  inferred: R  upper bound(s): javax.measure.Quantity<R>,javax.measure.Quantity<capture#1 of ?>  The big difference between the asType(Class) method and the new method signature is that a Speed speed = asType(Speed.class) call would throw a ClassCastException at runtime, while the new method signatures can not perform any verification. This is because of the way generic types are implemented in Java (by erasure). |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| @Keilw, I disagree. With the new method signature:  Quantity<Mass> timeToEvacuate = distanceKm.divide(speed);  the compiler does not produce anymore any error message or any warning, even at runtime. With the old method signature and a cast:  Mass timeToEvacuate = (Mass) distanceKm.divide(speed);  at least we get a ClassCastException. Or alternatively (if we provide the same method than Unit.asType(Class):  Quantity<Mass> timeToEvacuate = distanceKm.divide(speed).asType(Mass.class);  But I don't see why we would do the later if the (Mass) cast works. But at least the two last solutions perform a check at runtime, while the new method signatures remove **all** checks. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 12/Oct/14 ] |
| asType() showed to cause many problems under SE8. Especially with mixed operations involving type change. Have to check, if the BMIDemo and others exist for RI and SE port. We need both, at the end of the day (PDR) both implementations must also pass the TCK |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| Can you elaborate the problems with asType? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 12/Oct/14 ] |
| Just do a search (in Eclipse or another IDE) for "asType" in just the relevant project both under RI and SE port. E.g. UCUM has 38 usages of it in RI, but just one in SE. All other 37 were removed/replaced due to compiler errors when building the project  I just ported the BMIDemo: <https://github.com/unitsofmeasurement/uom-demos/blob/master/domain/health-se/src/main/java/tec/uom/demo/health/se/BMIDemoSE.java> There were no problems with it. Could you either push the example that gave you problems into one of the "demo" repositories or attach a patch here?  Furthermore, it made no difference with the old signature. Implementations are "allowed" to throw a ClassCastException, but this cast  Quantity<Energy> timeToEvacuate = (Quantity<Energy>) l.divide(metricSpeed);  works with divide() (which neither Leo nor anybody else had changed for now) in the old form. Using Energy instead of Time. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| Try the same BMIDemoSE example, unchanged, but with javac compiler source level set to 1.7 instead than 1.8. The compilation fails with the error shown above. Note that lot of JEE applications are still on Java 1.6 and very reluctant to upgrade, even after the end-of-life of that Java version. Those industrial accept to pay Oracle for JDK 1.6 commercial support.  But anyway, even if we stick to Java 1.8, the proposed change is just wrong as is defeat the generic type purpose. Again, usage of <?> is not necessarily a sign of bad design. It is just that a language try to scope with 90% of the usage, and we fall in the remaining 10%.  On asType, could you elaborate about what were the compiler errors? I'm absolutely sure that the can be solved. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| The cast to (Quantity<Energy>) is unsafe and I agree that they should be avoided. But this cast is not needed. Why not just cast to (Energy) instead, which is safe? Is it because the implementation fails to return instance of Energy? If so, then the problem is in the implementation, not in the API. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| Implementation are "allowed" to throw ClassCastException, but there is no way they can do that! This is because of type erasure. The only way to allow implementation to perform a check would be to provide a Class<R extends Quantity> argument in the method signature, which I would not recommend.  So on the implementation side, none of the signature (old or new) can perform a check. But the new signature gives a **FALSE** feeling of safety - it is **lying** to the developer. The old signature on the other hand was honest about the guarantees that it can not make. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 12/Oct/14 ] |
| That is up to the implementation, but most users prefer a generic "Factory" approach. Both Quantities and QuantityFactory (the latter on an API level at least as interface) will return Quantity<Energy> not {{Energy}, too.  Only implementations with every single class for every Quantity sub-type, e.g. TemperatureAmount (see <https://github.com/unitsofmeasurement/uom-lib/tree/master/si>) implement interfaces like Temperature. Looking at future prospect of Value Types (Valhalla) I am not sure, if we want to declare hundreds of thousands of these (also on a class base for now one reason why these concrete classes are final is they are similar to value classes, but the latter could only implement Quantity so they'd be closest to our enum example <https://github.com/unitsofmeasurement/unit-impl-enum>) especially in ME?  Anybody is free to do so, these implentations have to support the API, but most users and EG members seem to prefer a more generic approach with a factory to create Quantity instances.  Without strict stuff like Type Annotations we probably should not try to prevent apps from doing "stupid" things like casting to a wrong type, but a couple of experts like Otavio, Leo seem to be more comfortable with a typed return than having to cast, that is why the change was proposed. It does not prevent <?> either. As |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 12/Oct/14 ] |
| It forced to cast in any case which some developers feel uncomfortable with.  <https://github.com/unitsofmeasurement/uom-demos/blob/master/domain/health-se/src/main/java/tec/uom/demo/health/se/BMIDemoSE.java> shows, both versions of multiply() are accepted. It seems you prefer the <?> one, which I'd be equally fine with in general. On the other hand the correct  Quantity<Area> squareHeight = height.multiply(height);  instead of  Quantity<?> squareHeight2 = height.multiply(height);  or  Quantity<Area> squareHeight = (Quantity<Area>)height.multiply(height);  (before the signature change)  adds some convenience if you know what you're doing. I also would not clutter the API with some multiply(Quantity<C>, Class<C>) type of method. That would bring Java compiler oddities into an API that shall provide FUNCTIONAL results, whether you use it from Lambdas or not |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| A QuantityFactory could return Energy if the return type is Q rather than Quantity<Q>. But this is an other topic since this JIRA task is about the return type of Quantity.multiply and similar methods. Quantity factories are not involved here (at least on the API level).  Implementations do not need to provide single class for every Quantity sub-type. The java.lang.reflect.Proxy approach works well on JavaSE. For JavaME environments without reflection support, I presume that they would have only a subset of quantity types anyway, isn't it?  Developers may cast to the wrong type, but at least they know that they are doing something unsafe. However developers expect either a compiler warning or a ClassCastException at runtime. With the proposed change, they get none.  The idea of parameterized types is that if a code does not have any "unsafe cast" or "raw types" compiler warning, then it guaranteed that the code will never get object of the wrong type at runtime (he may get ClassCastException however). The proposed change breaks this contract. It is always risky to speak for someone else, but I would be ready to bet a lot that such huge hole would be considered unacceptable by API designers in the JDK team. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 12/Oct/14 ] |
| Keilw, it is not a matter of preference or convenience. **It is a matter of not lying to the developer**, which I'm 100% convinced would be considered unacceptable by JDK API designers. The parameterized types are supposed to **guarantee** that the returned object is of the declared type. If the API can not provide such guarantee, it shall not pretend that there is no unsafety. Either the user cast himself so he see the "unsafe cast" warning, or either he invoke a method like asType(Class) which is capable to perform a runtime check. If <?> is considered unconvenient, then we may consider removing completely parameterized types, but certainly not perverting them! |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 12/Oct/14 ] |
| As mentioned asType() got serious problems in Unit already. It would get even worse if added to Quantity. The return type of arithmetic operations must be compatible with what QuantityFactory or equivalent like the Quantities facade creates.  I just raised a pledge to refactor QuantityFactory but please discuss it there, not here, this thread is already quite extensive  Except for a compiler warning, that a cast must be applied (which at least Otavio considers bad practice) in the old signature, everything remains the same. If you chose <?> as return value there is and there was no warning. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 13/Oct/14 ] |
| The compiler warning is not about that a cast must be applied. The compiler warning is about the fact that the cast is unsafe. Hiding this fact is not a good practice, it makes it worst!  On other items:   * Can you explain explicitly (eventually in an other thread) what were asType(Class) problems? * Do we have a link to another thread about QuantityFactory, so we can fix it instead? |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 13/Oct/14 ] |
| Martin, you are right the new method just believe in the user. This way would be possible do something semantically wrong like this:  Quantity<Time> time = mass.multiply(length);  And with Erasable effect in Java, we cannot take care this semantic (Ok there is the Valhalla project, but this project could be not ready at time). However take care all combinations are not the goals of this JSR, but the JSR 275 (I bet).  I believe the discussion is not **technical**, but **strategic** to define the next steps and goals of this JSR. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Oct/14 ] |
| I had a brief discussion with Martin at least around 308 once. He said, it likely can't replace the aims of 275/363, but a combination seems possible. That is an area where near 100% semantic safety at least for particular domains could be reached, but 308 is totally untested. At the moment I can't say if it works for reference types, all its examples say "no".  Valhalla may not come before Java 9, nor is Jigsaw really going to help us with the modularity question (especially the way modules are cut it won't be a "Pick a single type inside a package") and the "stripped implementation" discussion depends on so many clogged up legal discussions in JSR 358 that I would bet on that as the very least. Beside there won't be a Jigsaw ME for the time being, Mark Reinhold confirmed when I asked the BOF.  At the end of the day we need to decide on what is the best compromise between type safety. Quantity is clear it provides "some" type safety, and even 308 Checkers claim, they cannot tell the difference between m^3 and m³ (it bizarrely proposes separate annotations for each of them, and there is some "Reverse-Polish" like annotation for prefixes, not to mention, unlike 363 those checkers will only work at Compile Time, there is no Runtime support, same for F# btw., so I'd say we have an advantage even if some mathematical formulas may contain an uncertainty or still force users to THINK ) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Oct/14 ] |
| See EG discussion and decision |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-61] [Hide Generics to avoid <?>](https://java.net/jira/browse/UNITSOFMEASUREMENT-61) Created: 02/Oct/14  Updated: 02/Oct/14  Resolved: 02/Oct/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-62~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-62) | Could we specialize Generics for oper... | Resolved | |  |
| **Tags:** | design, question |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The gerenics was created to solve cast's problem, this way it is not necessary do the cast.  In quantity some times is difficult to preview the answer, for example, when divide two quantities. This way is necessary to create the cast, but a new API with generics that does cast could means a bad API design. Maybe the best strategy is hide the Generic type, this way:  public interface Quantity {  }  public interface Time extends Quantity {  }  And this information about the quantity type will hide in the class:  {code:java}  public class DoubleQuantity {  private class<? Extends Quantity> type;  private Unit unit;  }  BTW: the operation in unit could be a State design pattern, but I will create another task to talk about this.  obs: I am not good with nomenclature, the idea is just the design **and not the name**. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Oct/14 ] |
| This would destroy all compile time type safety we got over the years;-/  JavaDoc of Quantity is very clear: >This interface extends Measurement to specify the quantitative property associated to a class through class parameterization and to provide some compile >time dimension consistency.  Having "some" compile time consistency (6 out of 8 methods in Quantity, as Martin hinted, there may be a flaw with inverse() that required us to add at least one more <?> because it's not physically and mathematically correct otherwise ) is better than having NONE as every other API out there. Especially those compatible with both Java ME and SE. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Oct/14 ] |
| See comment, the intention was good, but it would do the API more damage than good. JavaFX and other frameworks also use wildcards where necessary, e.g. change listeners, when you don't know the type of the new element or just look at Collection with various wildcards like containsAll().  This older book about Java 5 generics provides a nice overview of all sorts of Generics: <http://www.angelikalanger.com/GenericsFAQ/FAQSections/TypeArguments.html#FAQ103> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Oct/14 ] |
| If you want to avoid a cast e.g. in ThePerfectStorm example, all you need to do is:  Quantity<?> timeToEvacuate = l.divide(metricSpeed);  It is up to developers if they insist on a typed return value for further calculations, then they may need to cast, otherwise just stick with Quantity<?>. As mentioned in other comments, the return type may be a quantity that is new or not defined by SI. It is still valid and legitimate, e.g. kg/s but you can't pin it down with the Quantity sub-types defined by the API or RI, hence we need to provide this freedom to users to use the wildcard for new results of their choice |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 02/Oct/14 ] |
| I don't know, because, maybe hide this problem using:  Quantity<?> timeToEvacuate = l.divide(metricSpeed);  **Or**  Quantity timeToEvacuate = l.divide(metricSpeed);  Maybe using this way  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> divide(Quantity<T> that, Class<E> asType);  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> multiply(Quantity<T> that, Class<E> asType);  <T extends Quantity<T>> Quantity<T> inverse(Class<T> clazz);  This way, I can do this way:  Quantity<Speed> timeToEvacuate = length.divide(time, Speed.class);  We can verify and catch an exception if the operation cannot be released. This way we need to use cast in inverse. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Oct/14 ] |
| Please discuss further on the follow-up task 62 about specializing the generic arguments |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-60] [Liaise with MathML](https://java.net/jira/browse/UNITSOFMEASUREMENT-60) Created: 30/Sep/14  Updated: 07/Apr/15  Resolved: 07/Apr/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Trivial |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | xml |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| MVC Spec Lead Manfred Riem from Oracle mentioned MathML more or less by coincidence in the JSF F2F.  EG Members previously were involved in other markup languages like UnitsML, so we might look into possible synergies with MathML on the output/presentation level. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Apr/15 ] |
| Out of scope for RI/JSR. Possibly of interest for Tools <https://github.com/unitsofmeasurement/uom-tools/issues/9> or other UOM projects. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-59] [Remove "generic" multiply/divide operations from Quantity?](https://java.net/jira/browse/UNITSOFMEASUREMENT-59) Created: 29/Sep/14  Updated: 02/Oct/14  Due: 06/Oct/14  Resolved: 02/Oct/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, question |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Otavio suggested that we could remove the generic operations of Quantity (those with <?>), because it broadens the scope of the API too much. For example, it would have to take into account all variations possible for speed.  Me (Leonardo), Werner and Mohammed agreed on that, we'd like Jean-Marie's and Martin's opinions please |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 30/Sep/14 ] |
| Could you elaborate a little bit about why it would need to take in account every possible variations for speed? Isn't the implementation as simple as below?  Quantity<?> multiply(Quantity<?> multiplier) {  Number product = ... // Perform the product here, preserving the type if we can.  return quantityFactory.create(product, getUnit().multiply(multiplier.getUnit());  }  Of course, this assumes that the QuantityFactory implementation is capable to create the appropriate Quantity subclass for the given unit. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Oct/14 ] |
| After looking into the Unit interface and how comparable APIs like OSGi Measurement deal with that I don't agree we should remove it. <http://www.osgi.org/javadoc/r4v42/org/osgi/util/measurement/Measurement.html>  has exactly the same operators. mul(double) and mul(Measurement) (which in our case would be equivalent to Quantity).  The entire fabric of the JSR relies on multiply() or divide() methods in Unit and naturallly it's also a generic  Unit<?> multiply(Unit<?> that);  All the quantity implementations do is refer to this method via  return new DoubleQuantity(value / that.getValue().doubleValue(), getUnit().divide(that.getUnit()));  At most we run into formatting problems, e.g. ThePerfectStorm demo  252.6668058665553 km/h (FIVE)  Distance: 500 km  Time to evacuate: 1.9788907303639738 km/km/h  It should be just "h", so the formatting algorithm has room for improvement, but the divide() works. The demo requires a cast there. Whether or not the Unit method asType() can be applied here, too.  Unit<Speed> C = METRE.times(299792458).divide(SECOND).asType(Speed.class);  that is worth discussing. I don't think we do the standard effort any good if we removed it from the API.  We have very specialized test classes with methods like  public VolumeQuantity multiply(DistanceQuantity d1) {  AreaQuantity dq0 = convert(AreaUnit.sqmetre);  DistanceQuantity dq1 = d1.convert(DistanceUnit.m);  return new VolumeQuantity(dq0.units \* dq1.units, VolumeUnit.cumetre);  }  Some based on work by Paul Morrisson. These have explicit multiply() or divide() methods, but without more general API level methods we totally lose out on mathematic or scientific calculations. And a former EC rep for TOTVS just said yesterday, this is an area where Java losing a lot of ground to Python or other dynamic languages. |
| Comment by [dautelle](https://java.net/jira/secure/ViewProfile.jspa?name=dautelle) [ 01/Oct/14 ] |
| I agree with Martin and Werner there is no good reason to remove it.  We have to keep things as close as possible to the reality (real world). In the real world we can multiply any quantity by a coefficient (number) or by another quantity (area = lengh \* length). This should be reflected by the API.  Note: In the API I would have replaced the interface name Measurement with Measure (but this is another topic). |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 01/Oct/14 ] |
| I know, there is I good reason to these methods. But using as type, which guarantee I have? Modifying the Quantity to this way:  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> divide(Quantity<T> that, Class<E> asType);  <T extends Quantity<T>, E extends Quantity<E>> Quantity<E> multiply(Quantity<T> that, Class<E> asType);  What should happen when I use:  Length/Time and convert to Information?  The problem is the too long variations that there are in quantities and the implementation should take care all or does nothing, maybe next versions but not now. But If it is really important we can create the lib to do that, but not in the API. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Oct/14 ] |
| Well as mentioned OSGi Measurement provides arithmetics, and running a simple example like this:  Measurement m1 = new Measurement(10d, Unit.kg);  Measurement m2 = new Measurement(5d, Unit.s);    Measurement m3 = m1.mul(m2);  System.out.println(m3);  Measurement m4 = m1.add(m2);  System.out.println(m4);  The first operation returns 50.0 s\*kg while the second throws an {ArithmeticException}} " Cannot add kg to s". JSR 363 goes a bit further as the illegal addtion of kg and s is prevented at compile time, but there are situations where we can't predict the outcome. An operation like:  Measurement m5 = m1.div(m2);  System.out.println(m5);  returns 2.0 kg/s  This is perfectly legitimate, imagine the QS scenario as demonstrated by Gerrit yesterday in his Wearable session or any Fitbit, Strava and similar use case. Losing x kg walking an amount of time seems quite plausible, thus we must not judge the outsome as illegal or wrong. Even if the result may not have a distinct quantity. The information and (assuming we get our act together on formatting a bit more ) its visual representation are perfectly fine e.g. for a Fitness app. Thus we may add an extra method like Otavio suggested, OSGi Measurement also got public Measurement mul(double d,Unit u) but I am not sure, if that makes sense for API or implementation level, e.g. AbstractQuantity The general method should still remain, the output is valuable, and not in every case do people need to have a strict resulting quantity. UCUM is probably the best example, there is no need for any concrete quantity types as parsing and formatting is controlled by a large XML file, which may change over time |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 02/Oct/14 ] |
| And about the merits Methods divide and multiply?  Can we use generics to avoid cast? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Oct/14 ] |
| Well we do, but the outcome is never known. It could be a type that is just a general Quantity so you cannot do it without the wildcard here.  Please let's separate suggestions for e.g. an additional method with some "fancy" generic arguments or the class (similar to asType()) and possibly renaming Measurement to Measure (inspiration came largely from OSGi bundle, but there are other libraries like F# or ICU that call it Measure, too ) raising new tickets for that. If you need to link it to this one, feel free, but I guess it is best to close this one as the immediate action is not agreed on. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-58] [Consider a license change to BSD for RI/TCK](https://java.net/jira/browse/UNITSOFMEASUREMENT-58) Created: 19/Sep/14  Updated: 04/Nov/14  Resolved: 04/Nov/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200), [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-64~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-64) | Make JSR available to Eclipse Orbit | Resolved | |  |
| **Tags:** | external, license |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Contrary to discussions we had in January with Jim Wright, where he suggested Apache 2.0 was better than BSD (as JSR 275 also used) there have been issues between OSI/FSF and Apache considering Apache 2.0 incompabible with GPL: <http://www.apache.org/licenses/GPL-compatibility.html>  This affects large parts of the Java platform licensed under GPL+Classpath Exception.  While JSR 363 is not currently aiming at platform integration (there is less of a "platform" other than CLDC/MEEP in Java ME anyway) a future Java version like 9 or 10 could find some of it appealing. Either for Sensor support (on the SE side) or backing value objects as per "Valhalla" (whether that comes with Java 9 or 10 we shall see)  The license change to Apache 2.0 from earlier work based on 275 was decided by all EG members who contribute code and of course especially the Spec Leads. We can discuss it at the F2F, there will of course be a JSR 358 session next week involving those of us in the EC (Leo, Mohamed, myself and at least Bruno for SouJava)  Given it builds on top of Java SE 8+ the SE port is a piece which would be at least as important, if not more crucial than RI/TCK about this. Demos or other 3rd party projects could probably remain Apache. Especially one derived from a JBoss example is based on Apache already.  We have not published a milestone like EDR yet, so it would be a lot easier to do this now if we found it beneficial.  WDYT? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Sep/14 ] |
| As per JCP EC discussion on IP (JSR-358) any JSR that uses Apache license will be permitted from contributing to the Java platform. As soon as Java 10+ introduces not only Jigsaw but things like custom data types (Valhalla) there could be parts to enter the platform, so we don't want to prohibit that by using a license that is not acceptable by OpenJDK/Oracle.  Unit-API 0.6 and JSR-275 used BSD already, so we plan to relicense all relevant artifacts to BSD 2.  Unless any EG Members have serious objections, we'll do that within a week and also inform PMO. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 04/Nov/14 ] |
| As per <https://jcp.org/en/jsr/detail?id=363#updates> PMO also updated the detail page. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-57] [Latest Java 7 and 8 compilers break ME 8 compatibility](https://java.net/jira/browse/UNITSOFMEASUREMENT-57) Created: 19/Sep/14  Updated: 13/Dec/14  Resolved: 13/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | jdk8, timewait, unplanned | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Attachments:** | NB\_ME8\_run.txt |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-3~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-3) | Portability of RI | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-10~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-10) | Java 8 issues | Resolved | |  |
| **Tags:** | embedded, external, inconsistency, jdk |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Early Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| I found a serious problem with Java 7 and 8 (likely caused by some recent "security" fix or other changes see SE 8u20 breaking several apps)  Even building against JDK 1.5 with a Java 7 version 55 or later breaks the apps in Java ME 8 (Final emulator) see:  java.lang.UnsupportedClassVersionError   * java/lang/Class.invoke\_verify(), bci=0 * java/lang/Class.initialize(), bci=110 * java/lang/Class.forName(), bci=0 * com/sun/midp/main/CldcMIDletLoader.newInstance(), bci=1 * com/sun/midp/midlet/MIDletStateHandler.createMIDlet(), bci=63 * com/sun/midp/midlet/MIDletStateHandler.createAndRegisterMIDlet(), bci=16 * com/sun/midp/midlet/MIDletStateHandler.startSuite(), bci=25 * com/sun/midp/main/AbstractMIDletSuiteLoader.startSuite(), bci=38 * com/sun/midp/main/CldcMIDletSuiteLoader.startSuite(), bci=5 * com/sun/midp/main/AbstractMIDletSuiteLoader.runMIDletSuite(), bci=132 * com/sun/midp/main/AppIsolateMIDletSuiteLoader.main(), bci=26   JARs built with earlier JDK versions (up to Feb shortly before ME 8 went Final) still work, both API and the tailored Enum impl (where all SE traces are already gone)  We must build API, RI and special ME implementations (like enum-impl) with an appropriate JDK, ideally Java SE 6u45, the most recent Java 6 version. Optionally an older version of Java 7 might work.  Especially for CI servers (will create a separate ticket if the API on GeoAPI Maven was compromised and not working) this is important for API and all ME 8 compliant artifacts. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 19/Sep/14 ] |
| This is quite bizarre and worrying. At the moment a combination of Unit-API 0.5 and Enum-Impl 0.4 (both actually built with an EA/OpenJDK version of Java 8) work together in CLDC 8/ME 8.  Everthing built beyond that, especially with later versions of Java 8 consistently fails. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Sep/14 ] |
| All parts of the ME 8 SDK like MEEP JARs are built with JDK 7   * CLDC 8: Created-By: 1.7.0 (Sun Microsystems Inc.) * MEEP 8: Created-By: 1.7.0 (Oracle Corporation)   Even building Unit-API and Enum-Impl or any other (ME compliant) implementation against exactly the same JDK 1.7.0 fails under ME 8 SDK/emulator. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Sep/14 ] |
| Console log of NB running the app when error occurs |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 27/Sep/14 ] |
| Werner It's happened with many projects that included tomcat, groovy, Jboss Seam, Jboss Forge. The error was reported, just wait to new update. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Sep/14 ] |
| Still happens with ME SDK 8.1 |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 10/Oct/14 ] |
| I verified this to happen in current K70 binaries too... |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/Oct/14 ] |
| Thanks for checking. K70 is a product/library of V2COM, right? It's likely not the only one affected then which should build pressure on Oracle to provide a fix soon |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Dec/14 ] |
| According to the JCP Hackathon and Leo's effort with the latest ME 8.1 SDK this problem seems resolved. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-56] [Could we move AbstractQuantity into API/SPI?](https://java.net/jira/browse/UNITSOFMEASUREMENT-56) Created: 17/Sep/14  Updated: 26/Jul/15  Resolved: 26/Jul/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Duplicate** | | | | | is duplicated by | [~~UNITSOFMEASUREMENT-98~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-98) | Consider providing the "Units" class ... | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-145~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-145) | Move Bootstrap mechanism into API | Resolved | | is related to | [~~UNITSOFMEASUREMENT-13~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-13) | Do we need AbstractMeasurement in RI? | Closed | | is related to | [~~UNITSOFMEASUREMENT-121~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-121) | Introduce "Pluggable" Unit Systems an... | Resolved | | is related to | [~~UNITSOFMEASUREMENT-42~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-42) | Could we add AbstractSystemOfUnits to... | Closed | |  |
| **Tags:** | api, design, implementation, refactoring |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| If as intended, some of the factory methods went from AbstractQuantity into either QuantityFactory or a concrete class like BaseQuantity would it make sense to provide a general abstract base class like that in the API or at least SPI of the JSR instead of particular implementations?  Several JSRs do this, Collection API with  AbstractMap etc. Servlet or Portlet API with  GenericServlet or GenericPortlet.  Whether we call it AbstractQuantity hence keeping it abstract or something like GenericQuantity we may see. Different JSRs provide both as mentioned. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 17/Sep/14 ] |
| I believe no, the API just should define the contract. You are referring old APIs, nowadays almost always the best solution is the composition. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Sep/14 ] |
| Glad to hear, another example for old APIs (just took nearly a decade to add to Java 8 ) is JSR 310. Where API and RI are horribly entangled and almost every implementation of the anemic interfaces like Temporal have completely different behavior and pose high degree of duplication...  To avoid that a common base class in RI/SE port will likely stay, but I would be OK to keep them in each implementation. Let's hear other's thoughts. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 29/Sep/14 ] |
| Similiar to #42, let's keep the API clean of any implementation! |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 11/Jul/15 ] |
| I would like to reopen this for consideration as at least one or the other concrete (or abstract) class must be factored into the API as a single point of access to the SPI/API.  Look at CDI it got a CDI class in the SPI package and there is at least one other class Unmanaged since 1.1.  The SPI and its package are optional, so it's a default implementation for SystemOfUnits which works the same way in SE and ME, so this would make it more common here (for pluggable Unit Systems) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 26/Jul/15 ] |
| There are a few implementation specific references, e.g. to AbstractUnit in AbstractSystemOfUnits, so moving it to API does not make sense in this form. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-55] [Remove generic V from Measurement](https://java.net/jira/browse/UNITSOFMEASUREMENT-55) Created: 17/Sep/14  Updated: 29/Sep/14  Resolved: 18/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | api, design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| After Measurement had most numeric methods and values delegated to the Quantity subtype, it seems less useful to keep a generic <V> argument there.  Where it makes sense is for Quantity to extend ValueSupplier<Number> but as described in recent JavaDoc for Measurement its concrete classes are free to implement ValueSupplier<DESIRED\_TYPE> to match the behavioral pattern of Quantity kind of like extending or implementing Comparable, Serializable, etc. but the base interface does not enforce it, so <V> seems redundant for Measurement. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 17/Sep/14 ] |
| [DONE] <https://github.com/unitsofmeasurement/unit-api/pull/11>  <https://github.com/unitsofmeasurement/unit-ri/pull/24> |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 17/Sep/14 ] |
| PR: to solve this design: <https://github.com/unitsofmeasurement/unit-api/pull/11> <https://github.com/unitsofmeasurement/unit-ri/pull/24> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Sep/14 ] |
| Unfortunately the change seriously broke the API;-( |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Sep/14 ] |
| Please refrain from @Override for places where interfaces are implemented. This may have started working from Java SE 8 on or so, but the API is backward compatible to Java 5 (also see this year's JavaOne Embedded Challenge winner Lhings: <https://github.com/lhings/java_lhings_library> ) which is why the @Override breaks build in some cases. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 17/Sep/14 ] |
| @kelliw I runned the test and is ok the API. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Sep/14 ] |
| SE port has not applied the changes yet. Compiler gets irritated by the @Override, so let's not use that please, the API needs to be Java 5 compatible, that is why it causes compile errors. Maven seems to ignore that if run against Java 7 or 8, which looks more like a Maven problem, the Java 5/6 code must not have @Override for implementing methods. So  @Override  toString()  is OK, but  @Override  getUnit()  isn't. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Sep/14 ] |
| After resolving @Override annotation problem it looks OK |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-54] [split funcionality in AbstractQuantity](https://java.net/jira/browse/UNITSOFMEASUREMENT-54) Created: 17/Sep/14  Updated: 17/Sep/14  Resolved: 17/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) | **Assignee:** | [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) |
| **Resolution:** | Invalid | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The proposal is, remove the factory's methods in AbstractQuantity and create the QuantityFactory:  I saw some advantagens:  Hide the implementations Just show the API the client don't need know the implementation Single responsability to factory - now one class will do that the AbstractQuantity follows the SOLID principle  So:  Quantity<Time> day = QuantityFactory.of(10, SI.DAY); |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Sep/14 ] |
| Sorry but part of it is clearly broken. Can we try to rephrase it.  AbstractQuantity has a place in the API and it is not a "factory" for quantities, but actual implementation of the Quantity interface. The name of the class shouldn't be misleading.  Either keep something like   * BaseQuantity * GenericQuantity (see Servlet JSR ) ...   using an of() method along the lines of what AbstractQuantity uses right now. AbstractQuantity (in Servlet-API the most common base type is called GenericServlet, specialized e.g. by HttpServlet, so another name could apply, but the common first level implementation shall be preserved for compatiblity of other types) and saving a lot of boilerplate code like holding the Unit, etc.  If QuantityFactory was to replace BiFactory on an API level, there is no place for the same name in the RI or elsewhere either btw.  Right now we have:  QuantityFactory qf = QuantityFactory.getInstance(Time.class);  Quantity<Time> day = qf.create(10, SI.DAY);  Following the pattern of MEEP 8 with methods like  EventManager ssm = EventManager.getInstance();  The getInstance() method is rather common and for a singleton (as returned by QuantityFactory) while of() is used in cases of a non-singleton factory result. It is a small detail, more importantly neither QuantityFactory.getInstance() nor a hypothetical QuantityFactory.of() must return Quantity, that's up to the create() method following the Factory Pattern.  We already have a task for renaming/rephrasing BiFactory or pulling QuantityFactory into the API, so I am inclined to reject that as duplicate unless there is a strong rephrase of its objective. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Sep/14 ] |
| Beside this is only for RI, if there are SE specific implementation details, please let's use GitHub for now |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 17/Sep/14 ] |
| @keliw The proposal is not delete the AbstractQuantity, just remove the factory`s method. To get more OO design. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-53] [Measurement extends Comparable](https://java.net/jira/browse/UNITSOFMEASUREMENT-53) Created: 17/Sep/14  Updated: 29/Sep/14  Resolved: 17/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) | **Assignee:** | [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Status Whiteboard:** | Quantity interface extends Comparable, this way will possible:  List<Time> list = Arrays.asList(Quantyty.factory(10, SI.HOUR), Quantyty.factory(100, SI.MINUTE), Quantyty.factory(2, SI.DAY));  list.sort()  The order will:  100 minutes 10 hours 2 days |
| **Tags:** | design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Sep/14 ] |
| As mentioned, to allow all Measurement types including Quantity, we should put that on top of Measurement, not just Quantity |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 17/Sep/14 ] |
| Ok, I agree with you. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 17/Sep/14 ] |
| Done:  <https://java.net/jira/browse/UNITSOFMEASUREMENT-53> |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 17/Sep/14 ] |
| I think that only Quantity should be Comparable, because Measurement can be anything. What means "comparable" for a vector? We could decide that we compare length, but this is arbitrary. And what means "comparable" for a range (do we compare min or max?), or for a raster? |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 18/Sep/14 ] |
| You are right. Another problem is, there isn't getValue() method in Measurement interface. And I believe that no make sense. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 18/Sep/14 ] |
| There is no getValue() in Measurement on intend, because it does not make sense for every kind of measurement. Taking again the example of a Range object, if someone decide that measurement can be a range, a getValue() does not apply (they have getMin() and getMax() instead. The range can hardly be Comparable neither (not impossible, but require arbitrary choices).  I think that Measurement should:   * Contains nothing else than a getUnit() method. * Is **not** comparable (the choice to implement Comparable is left to sub-types). * Have an extensive javadoc explaining what we mean by "measurement". |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Sep/14 ] |
| There is no reason Comparable needs to use getValue() in all implementing classes. Unfortunately we may have to push that to implementing classes like AbstractQuantity.  As there is a clash with enums implementing either Measurement or Quantity based on Comparable.  A real live example case ShirtSizeEnum is broken and made impossible if Comparable is put before Measurement or Quantity I'm afraid  public enum ShirtSizeEnum implements Measurement<ShirtSize>  No longer works as the Measurement (or Quantity) signature with Comparable<Measurement<Q>> and Enum also implementing Comparable collide. I'm afraid we have to mix-in Comparable on an implementation level, e.g. AbstractQuantity. That also allows leaving it from Measurement as Martin suggested, same with Quantity, as there is no reason not to implement a Quantity via an Enum either. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 18/Sep/14 ] |
| Ok, I am removing, so |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Sep/14 ] |
| Just doing in an outgoing change to API, so please do changes only in RI/SE for now. If we agreed and documented that Quantity must not be implemented by an Enum we may consider it for Quantity.  WDYT? |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 18/Sep/14 ] |
| Yes. I will do it with Lambda and Stream in SE implementation and Comparable in RI |

|  |  |
| --- | --- |
| **[Discuss \*Supplier functional interfaces](https://java.net/jira/browse/UNITSOFMEASUREMENT-50)** ([UNITSOFMEASUREMENT-50](https://java.net/jira/browse/UNITSOFMEASUREMENT-50)) [UNITSOFMEASUREMENT-52] [Look at ValueSupplier](https://java.net/jira/browse/UNITSOFMEASUREMENT-52) Created: 06/Sep/14  Updated: 06/Oct/14  Resolved: 06/Oct/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Unlike other 2 ValueSupplier is used throughout implementations, so it sounds reasonable to keep it.  Also a test class <https://github.com/unitsofmeasurement/unit-api/blob/master/src/test/java/javax/measure/test/StringMeasurement.java>  highlights, how particular Measurement implementations may use the same unified method signature to correspond with Quantity (and hence via ValueSupplier the code does not have to care, what interfaces it implements, as long as a getValue() method exists, a bit like Serializable but more than a simple tagging interface ) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 06/Oct/14 ] |
| ValueSupplier is used by Quantity and various RI elements especially UnitConverter implementations. It also offers Measurement implementations a behavior compatible with Quantity where a getValue() method is required. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-51] [Use distinct namespace for JSR and other projects](https://java.net/jira/browse/UNITSOFMEASUREMENT-51) Created: 25/Aug/14  Updated: 26/Aug/14  Resolved: 26/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200), [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-40~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-40) | Do we want to keep the unitsofmeasure... | Resolved | |  |
| **Tags:** | design, domain |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Otavio and I refactored all relevant projects/repositories using "tec.uom". Given historically the API was called "unit-api" and there is no real need to change that (the "javax.measure" package was used as in JSR-275, it also is used by several projects, so it makes a transition easier)  Could we follow a pattern of naming RI and TCK accordingly, similar to e.g. JSR-354 ("money-\*") with "unit-api", "unit-ri" and "unit-tck". As there are 2 equal namespaces available I suggest using "tec.units" for RI and TCK including the "unit-" prefix, while all other projects seem great with "uom-\*". Along the lines of "javamoney-" for 354. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-50] [Discuss \*Supplier functional interfaces](https://java.net/jira/browse/UNITSOFMEASUREMENT-50) Created: 25/Aug/14  Updated: 06/Oct/14  Resolved: 06/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-32~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-32) | Remove functional interfaces that are... | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-52](https://java.net/jira/browse/UNITSOFMEASUREMENT-52) | [Look at ValueSupplier](https://java.net/jira/browse/UNITSOFMEASUREMENT-52) | Sub-task | Resolved | keilw | |  |
| **Tags:** | design, question |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This sub-task is entirely for the 3 Supplier like interfaces:   * Nameable * UnitSupplier * ValueSupplier |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 05/Sep/14 ] |
| As we talked in the August call, if we can't bring a use case that shows value in the Interface themselves, we should remove them.  For now, we don't have any use case that justifies the existence of these interfaces. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 05/Sep/14 ] |
| I agree. Those interfaces are not needed for lambdas and I'm not aware of a use case where Nameable, UnitSupplier or ValueSupplier would be used in isolation (i.e. used instead of Unit, UnitSystem or Measurement). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 06/Sep/14 ] |
| Especially ValueSupplier is used across all implementations. Requires a more detailed look at it.  The others currently seem only used in the places discussed earlier today. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 06/Sep/14 ] |
| There's a case especially for ValueSupplier since it allows Measurement implementations to use it (like other parts of RI and SE impl already do) for a standard behavior like Quantity, but with different return types.  Java SE 8 Supplier is not a real alternative, as the only way it would work for all these (at least in the SE implementation ) is if these classes all used a method named get() that's the only one the out-of-the box Supplier allows. And it would be quite confusing next to methods like getUnit() etc. what get() actually stands for.  IMHO leaving all of these getValue() sounds more understandable. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-49] [Specialize ConversionOperator](https://java.net/jira/browse/UNITSOFMEASUREMENT-49) Created: 25/Aug/14  Updated: 28/Sep/14  Resolved: 28/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-32~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-32) | Remove functional interfaces that are... | Resolved | |  |
| **Tags:** | design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Remaining a functional (single method) interface, the current ConversionOperator with a signature like public interface ConversionOperator<T,R> could be specialized into something like: public interface MeasurementConverter<Q, R> with a functional method: to(Unit<Q> unit)  (in fact, a single <Q> signature may work, if the return value can always be assumed as Measurement<Q>) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| As JSR 354 is not yet final, let's look at the final 310. There is TemporalAdjuster (<http://docs.oracle.com/javase/8/docs/api/java/time/temporal/TemporalAdjuster.html>) Temporal adjustInto(Temporal temporal) and the more generic TemporalQuery (<http://docs.oracle.com/javase/8/docs/api/java/time/temporal/TemporalQuery.html>) R queryFrom(TemporalAccessor temporal)  Aside from parsing (which I don't want to cover here) we have 2 similar requirements, one to transform a unit into another unit (which is where UnitTransformer works with a single functional transform(UnitConverter method. The other (the name was always base on the Fowler suggestion, it could also be something like convert() or as the Node.js framework MCO calls it transform() or maybe Stephen Colebourne would suggest adjustInto() ) ConversionOperator with a single to() method, converting Measurement/Quantity objects into other compatible units. There is a need for these as 310 has the more generalistic (especially with the Query) ones in the "temporal" API sub-package.  For QuantityFactory there's a similar task, which I guess we have a bit of an understanding as to moving a more specific interface into the API, so different implementations may use it accordingly. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 27/Sep/14 ] |
| I agree. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 27/Sep/14 ] |
| Done:  <https://github.com/unitsofmeasurement/unit-api/pull/19> |

|  |  |
| --- | --- |
| **[Make API available to Snapshot Maven repository of GeoAPI](https://java.net/jira/browse/UNITSOFMEASUREMENT-41)** ([UNITSOFMEASUREMENT-41](https://java.net/jira/browse/UNITSOFMEASUREMENT-41)) [UNITSOFMEASUREMENT-48] [Make Parent POM available on Maven Snapshot repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-48) Created: 15/Aug/14  Updated: 25/Aug/14  Resolved: 25/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-40~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-40) | Do we want to keep the unitsofmeasure... | Resolved | |  |
| **Tags:** | deployment, maven, server |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Looking at the RI for the CDI Spec, like many other JBoss projects it contains multiple POMs  GroupId ArtifactId Latest Version Updated Download org.jboss.weld weld-core-bom 2.2.4.Final all (75) 18-Jul-2014 pom  org.jboss.weld weld-core-parent 2.2.4.Final all (75) 18-Jul-2014 pom  org.jboss.weld weld-api-parent 2.2.SP3 all (39) 07-Jul-2014 pom  org.jboss.weld weld-api-bom 2.2.SP3 all (39) 07-Jul-2014 pom  org.jboss.weld weld-parent 28 all (27) 17-Jun-2014 pom  These are just the POM projects for "weld-core", the main RI project, defining 2 POMs (BOM and parent) by itself, as well as having 2 more from the API and an overall "weld-parent". That is currently the only master POM we declare for UnitsOfMeasurement, and even if it was to shift to java.net the master POM of the project remains.  Could we publish that master pom on the Snapshot repo? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 25/Aug/14 ] |
| 0.7-SNAPSHOT deployed on <http://maven.geotoolkit.org/> |

|  |  |
| --- | --- |
| **[Remove functional interfaces that are out of JSR-363 scope](https://java.net/jira/browse/UNITSOFMEASUREMENT-32)** ([UNITSOFMEASUREMENT-32](https://java.net/jira/browse/UNITSOFMEASUREMENT-32)) [UNITSOFMEASUREMENT-47] [Merge Converter and UnitConverter](https://java.net/jira/browse/UNITSOFMEASUREMENT-47) Created: 15/Aug/14  Updated: 15/Aug/14  Resolved: 15/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Another sub-task, this was analyzed and confirmed already. |

|  |  |
| --- | --- |
| **[Remove functional interfaces that are out of JSR-363 scope](https://java.net/jira/browse/UNITSOFMEASUREMENT-32)** ([UNITSOFMEASUREMENT-32](https://java.net/jira/browse/UNITSOFMEASUREMENT-32)) [UNITSOFMEASUREMENT-46] [Remove MinimumSupplier and MaximumSupplier](https://java.net/jira/browse/UNITSOFMEASUREMENT-46) Created: 15/Aug/14  Updated: 15/Aug/14  Resolved: 15/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Splitting this into smaller sub-tasks, as 2 of them have been moved, the others need more clarification, see the "Multi-Supplier" dilemma where even under SE the same interface must not be used more than once |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-45] [UnitConverter.getCompoundConverters() should not return null](https://java.net/jira/browse/UNITSOFMEASUREMENT-45) Created: 14/Aug/14  Updated: 17/Sep/14  Resolved: 17/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-44~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-44) | Rename UnitConverter.getCompoundConve... | Resolved | |  |
| **Tags:** | javadoc |
| **Sprint:** | August, JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Current contract of UnitConverter.getCompoundConverters() said: "Returns the chain of fundamental converters making up this converter or null if this converter is a fundamental converter". I propose to change as below:  Returns the chain of fundamental converters making up this converter. If this converter is a fundamental converter, then this method returns a singleton containing only this.  The above admittedly makes implementation slightly more difficult, but seems logically more consistent to me. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Aug/14 ] |
| It would affect other implmementations, and since we're not based on Java SE 8 we cannot apply a "default" body, so JavaDoc seems the best place so clarify that |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-44] [Rename UnitConverter.getCompoundConverters() to getConversionSteps()](https://java.net/jira/browse/UNITSOFMEASUREMENT-44) Created: 14/Aug/14  Updated: 07/Sep/14  Resolved: 07/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-45~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-45) | UnitConverter.getCompoundConverters()... | Resolved | |  |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The UnitConverter interface contains a getCompoundConverters() method. I think that the method name is not appropriate. If I compare to similar API like ISO 19111, I see the word "compound" used for the **result** of composition or concatenation. This is not the intend of this method, which rather returns the **elements** of the composition. I propose to pickup an other name, for example:   * getConversionSteps() * or getSingleElements() |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Aug/14 ] |
| Could be explored during a Hackathon, e.g. at JavaOne. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 05/Sep/14 ] |
| As we talked in August call, let's rename it to getConverstionSteps(). |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 05/Sep/14 ] |
| Comparison with ISO 19111 international standard (Spatial referencing by coordinates). A document identical to ISO 19111 is [available for free](http://portal.opengeospatial.org/files/?artifact_id=39049) by the Open Geospatial Consortium (OGC). This is because this international standard is a join ISO/OGC effort.  **Table 6 (page 24): Compound CRS:** A coordinate reference system describing the position of points through two or more independent single coordinate reference systems.  The word "compound" is used here for designating the "big" object, which is composed of smaller objects. By contrast, the current UnitConverter.getCompoundConverters() seems to use the word "compound" for the small objects, which is not consistent with ISO 19111.  **Table 46 (page 52): Concatenated operation:** An ordered sequence of two or more single coordinate operations.  (note: a "coordinate operation" is a coordinate conversion or transformation - this is directly equivalent to our UnitConverter). If we accept ISO 19111 as a source of inspiration, "concatenated unit" would be preferable to "compound unit" for the result of UnitConverter.concatenate(UnitConverter).  For getting the components of a concatenated operation, I though that the method name was getOperationSteps() but I just checked and it appears to be getCoordOperations(). However, for JSR-363 I think that getConversionSteps() would be clearer. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-43] [Should we rename "service" to "spi"?](https://java.net/jira/browse/UNITSOFMEASUREMENT-43) Created: 13/Aug/14  Updated: 10/Sep/14  Resolved: 10/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-42~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-42) | Could we add AbstractSystemOfUnits to... | Closed | |  |
| **Tags:** | descision, design, question |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| So far the "service" package defines interfaces primarily in an OSGi sense, also usable by Java "ServiceLoader" mechanisms in theory. It is more or less what many JSRs or similar APIs consider an SPI.  See e.g. Log4J 2 with a cleaner separation of API and implementation, it also provides an SPI package including e.g. some abstract base classes for loggers, etc., too. A similar SPI package could enable a few similar offerings here, especially for cases like AbstractSystemOfUnits |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 05/Sep/14 ] |
| As we talked on August call, let's rename it so that it's more consistent with other libraries. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-42] [Could we add AbstractSystemOfUnits to the API/SPI?](https://java.net/jira/browse/UNITSOFMEASUREMENT-42) Created: 13/Aug/14  Updated: 29/Sep/14  Resolved: 29/Sep/14 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-43~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-43) | Should we rename "service" to "spi"? | Resolved | | is related to | [~~UNITSOFMEASUREMENT-56~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-56) | Could we move AbstractQuantity into A... | Resolved | |  |
| **Tags:** | design, question |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The AbstractSystemOfUnits base class in the RI and alternate implementations like SE 8 port are absolutely identical.  Right now this abstract base class has no reference to any concrete RI or implementation-specific type. It simply eases the usage of the SystemOfUnits interface and could cater in a direction, Anatole suggested without forcing any concrete implementation onto people. The name is somewhat flexible, but as long as concrete unit systems on RI level like SI, US, Imperial, etc. extend it, that name makes more sense than e.g. calling it Units.  It would not increase the API size too much it seems. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 27/Sep/14 ] |
| IMHO: Should be stay in just RI, because the API don't need know anything about the implementation. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 29/Sep/14 ] |
| I agree with Otavio |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-41] [Make API available to Snapshot Maven repository of GeoAPI](https://java.net/jira/browse/UNITSOFMEASUREMENT-41) Created: 12/Aug/14  Updated: 25/Aug/14  Resolved: 20/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Cloners** | | | | | clones | [~~UNITSOFMEASUREMENT-34~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-34) | Make API available to public Maven re... | Resolved | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-26~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-26) | Set up Jenkins job on GeoAPI | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-48](https://java.net/jira/browse/UNITSOFMEASUREMENT-48) | [Make Parent POM available on Maven Sn...](https://java.net/jira/browse/UNITSOFMEASUREMENT-48) | Sub-task | Resolved | desruisseaux | |  |
| **Tags:** | Maven, deployment |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| GeoAPI maintains at least one Snapshot repository <http://maven.geotoolkit.org/>  see <http://www.geoapi.org/maven.html>  It would be nice to deploy Snapshot releases of the API, possibly other artifacts (including master POM of unitsofmeasurement, etc.) especially the RI, there too. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 14/Aug/14 ] |
| Done for unit-api. For the RI (Reference Implementation), we still have the issue of its dependency to an other project, namely org.unitsofmeasurement:unitsofmeasurement-parent:pom:0.7-SNAPSHOT. It seems to me that creating a Jenkins job for many tiny projects would be an overkill (it increases the perceived complexity of all the network of projects, and the maintenance effort). What about consolidating some projects as modules in a single multi-modules Maven project? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Aug/14 ] |
| Having a parent POM for established Open Source projects is a sign of good practice, not overkill  See JCache, which has a Parent POM even for the TCK alone: <http://search.maven.org/remotecontent?filepath=javax/cache/tck-parent/1.0.0/tck-parent-1.0.0.pom> or JavaMoney: <http://search.maven.org/#search%7Cga%7C1%7Corg.javamoney>  I tried detaching the RI project from its parent pom and it would be a horrible mess, dozens, more Hundreds of Lines of duplicate code for the RI alone, which all other projects require (and whether you like to deploy it there, those using it will need that parent POM just like others, e.g. for java.net: <http://repo.maven.apache.org/maven2/net/java/jvnet-parent/1/jvnet-parent-1.pom>  Or take Apache SIS, it uses the Apache Master POM: <parent> 29 <groupId>org.apache</groupId> 30 <artifactId>apache</artifactId> 31 <version>14</version> <!-- See the "TODO" in this file. --> 32 </parent>  That Parent POM (we also use another version at DeviceMap) is ~400 lines. Even if some entries may be overridden, that saves individual POMs and projects like SIS most of these 400 lines. Same for UOM, the parent POM is currently of similar size. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Aug/14 ] |
| Related task (RI/Parent POM) is still open, but for API it seems fixed |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 24/Aug/14 ] |
| The proposal was not to not have a parent pom.xml, but to not create a source code repository for only that file. The parent pom.xml could have been defined at the root of the Reference Implementation (RI) source code repository instead, and the RI itself as a sub-module. Other demo or example codes could have been other sub-modules.  Having to checkout many source code repositories is understandable for very large projects, but even in those cases the projects often provide some way (e.g. the Mercurial Forest on OpenJDK) for making life easier for developers and Jenkins administrators. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| I'm afraid that would be a bad duplication, too. See the SE project by Otavio and all others having "uom-parent" define what they need to know, including Snapshot repo.  If we defined that for RI, then (though it is a bad practice, but may have been done e.g. by IBM for JBatch) a structural root for RI and TCK would be imaginable but dirty and redundant. It will always duplicate what's in the parent POM ;-/ |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| JCache (JSR107) is a very minimalistic project, there POMs are disconnected, see <https://github.com/jsr107/RI/blob/master/pom.xml>  However, the "full" implementation EHCache like all other proper Open Source projects also has a parent POM. <http://search.maven.org/remotecontent?filepath=org/ehcache/ehcache-jcache-parent/1.0.0/ehcache-jcache-parent-1.0.0.pom>  We would not have to create that parent pom on a regular basis in a Jenkins job, but it must exist in Maven repositories involved. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-40] [Do we want to keep the unitsofmeasurement.org namespace?](https://java.net/jira/browse/UNITSOFMEASUREMENT-40) Created: 11/Aug/14  Updated: 25/Aug/14  Resolved: 25/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200), [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202), [TCK](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15201) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-48~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-48) | Make Parent POM available on Maven Sn... | Resolved | | blocks | [~~UNITSOFMEASUREMENT-51~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-51) | Use distinct namespace for JSR and ot... | Resolved | | blocks | [~~UNITSOFMEASUREMENT-35~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-35) | Make RI and related libraries availab... | Closed | |  |
| **Tags:** | design, domain, question |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Jean-Marie recently flagged the soon expiring domain for www.unitsofmeasurement.org he registered for the past 4 years or so.  While the API for the new JSR may use the namespace javax.measure, that does not apply to any other artifact, including RI, TCK, libraries or demos. Currently for Maven or Java packages they use org.unitsofmeasurement.  If we intend to do so in future, somebody (ideally a Spec Lead, mostly because if we keep the dommain, MR ownership and responsibility also stays with Spec Leads, while the job of the EG is more quiet then) may want to keep registering that domain, at least for some time.  A fairly "tiny" JSR that's practically gobbled up by CDI now was 330. It shares project space with the old Unit-API (would start on GitHub now ) and also has at least one or the other Google Group like we do. However, while the minimalistic TCK uses former domain "atinject.org" <https://code.google.com/p/atinject/source/browse/#svn%2Ftrunk%2Ftck%2Forg%2Fatinject%2Ftck> this domain is now for sale by GoDaddy and no longer owned by Bob Lee. Others still keep them alive, at least for some time.  If we wanted to move completely away from the .org domain, java.net seems a very plausible altarnative, see projects like TrueCommons: <https://truecommons.java.net/> One can find many Maven modules, including pure POMs for it here: <http://search.maven.org/#search%7Cga%7C1%7Cg%3A%22net.java.truecommons%22>  WDYT? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Aug/14 ] |
| I registered a domain [http://uom.technology](http://uom.technology/) and (potentially for RI usage) [http://uom.si](http://uom.si/). While they are "external domains" to the registrar and may not be so easy for web hosting, I also claimed "uom.tec" and "units.tec" for initially 5 years.  Being very short and distinctive, I suggest using either of these for all relevant artifacts. "uom.si" may work for things like RI or SI only components, otherwise the general "uom.tec" sounds best.  WDYT? |

|  |  |
| --- | --- |
| **[Website](https://java.net/jira/browse/UNITSOFMEASUREMENT-36)** ([UNITSOFMEASUREMENT-36](https://java.net/jira/browse/UNITSOFMEASUREMENT-36)) [UNITSOFMEASUREMENT-39] [Add new content from recent events](https://java.net/jira/browse/UNITSOFMEASUREMENT-39) Created: 10/Aug/14  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.0](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16917) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | content, events, web |
| **Sprint:** | August, Q1/15, Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Add new content like pics and reports from recent events like ECF, DemoCamps or DWX. |

|  |  |
| --- | --- |
| **[Website](https://java.net/jira/browse/UNITSOFMEASUREMENT-36)** ([UNITSOFMEASUREMENT-36](https://java.net/jira/browse/UNITSOFMEASUREMENT-36)) [UNITSOFMEASUREMENT-38] [Improve the favicon](https://java.net/jira/browse/UNITSOFMEASUREMENT-38) Created: 10/Aug/14  Updated: 29/Sep/14  Resolved: 29/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277), [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | content, icon, web |
| **Sprint:** | August, Q1/15, Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Improving the favicon (still has a few non-transparent pixels, etc.)  Possibly an option for an icon-choice for the project. |

|  |  |
| --- | --- |
| **[Website](https://java.net/jira/browse/UNITSOFMEASUREMENT-36)** ([UNITSOFMEASUREMENT-36](https://java.net/jira/browse/UNITSOFMEASUREMENT-36)) [UNITSOFMEASUREMENT-37] [Add Calendar page](https://java.net/jira/browse/UNITSOFMEASUREMENT-37) Created: 10/Aug/14  Updated: 10/Aug/14  Due: 01/Aug/14  Resolved: 10/Aug/14 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | 0 minutes | | |
| **Time Spent:** | 2 days | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | calendar, content, web |
| **Sprint:** | August, Q1/15, Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Adding a Google Calendar for Units of Measurement |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Aug/14 ] |
| Done, see <http://unitsofmeasurement.github.io/pages/calendar.html> |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-36] [Website](https://java.net/jira/browse/UNITSOFMEASUREMENT-36) Created: 10/Aug/14  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277), [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.0](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16917) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | 0 minutes | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | 2 days | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-37](https://java.net/jira/browse/UNITSOFMEASUREMENT-37) | [Add Calendar page](https://java.net/jira/browse/UNITSOFMEASUREMENT-37) | Sub-task | Closed | keilw | | [UNITSOFMEASUREMENT-38](https://java.net/jira/browse/UNITSOFMEASUREMENT-38) | [Improve the favicon](https://java.net/jira/browse/UNITSOFMEASUREMENT-38) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-39](https://java.net/jira/browse/UNITSOFMEASUREMENT-39) | [Add new content from recent events](https://java.net/jira/browse/UNITSOFMEASUREMENT-39) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-135](https://java.net/jira/browse/UNITSOFMEASUREMENT-135) | [Migrate from unitsofmeasurement.org](https://java.net/jira/browse/UNITSOFMEASUREMENT-135) | Sub-task | Resolved | desruisseaux | |  |
| **Tags:** | content, web |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | August, Q1/15, Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| An umbrella story for the project website(s) |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-35] [Make RI and related libraries available to public Maven repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-35) Created: 08/Aug/14  Updated: 25/Jun/15  Resolved: 25/Jun/15 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-40~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-40) | Do we want to keep the unitsofmeasure... | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-83](https://java.net/jira/browse/UNITSOFMEASUREMENT-83) | [Make RI and libraries available on Bi...](https://java.net/jira/browse/UNITSOFMEASUREMENT-83) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-84](https://java.net/jira/browse/UNITSOFMEASUREMENT-84) | [Make RI available on MavenCentral](https://java.net/jira/browse/UNITSOFMEASUREMENT-84) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-123](https://java.net/jira/browse/UNITSOFMEASUREMENT-123) | [Make libraries available on MavenCentral](https://java.net/jira/browse/UNITSOFMEASUREMENT-123) | Sub-task | Closed | keilw | |  |
| **Tags:** | Maven, deployment |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Q1/15, Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Se API task, the RI and related artifacts (at least parent POM) should also be published to a public Maven repository.  See JSR 354 RI "Moneta" and its parent POM <http://search.maven.org/#search%7Cga%7C1%7Cjavamoney>  A large number of JSRs between EDR and Final release do this and it'll increase visibility and access to the JSR from EDR stage on. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-34] [Make API available to public Maven repositories](https://java.net/jira/browse/UNITSOFMEASUREMENT-34) Created: 08/Aug/14  Updated: 29/Dec/14  Resolved: 29/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886), [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17107) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Cloners** | | | | | is cloned by | [~~UNITSOFMEASUREMENT-41~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-41) | Make API available to Snapshot Maven ... | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-75](https://java.net/jira/browse/UNITSOFMEASUREMENT-75) | [Make API available on Bintray.com](https://java.net/jira/browse/UNITSOFMEASUREMENT-75) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-76](https://java.net/jira/browse/UNITSOFMEASUREMENT-76) | [Make API available on MavenCentral](https://java.net/jira/browse/UNITSOFMEASUREMENT-76) | Sub-task | Resolved | keilw | |  |
| **Tags:** | Maven, deployment |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At least in EDR stage, the EDR version (0.7 as of now) of the JSR 363 API should be made available on public Maven repositories (MavenCentral, etc.)  A large number of JSRs between EDR and Final release do this and it'll increase visibility and access to the JSR from EDR stage on. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Dec/14 ] |
| Done as of EDR. Further stages may require deployment as part of a stage task again. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-33] [Remove javax.measure.util Package](https://java.net/jira/browse/UNITSOFMEASUREMENT-33) Created: 05/Aug/14  Updated: 08/Aug/14  Resolved: 08/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-31~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-31) | Remove javax.measure.util.TimedData | Resolved | | depends on | [~~UNITSOFMEASUREMENT-30~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-30) | Remove javax.measure.util.Range | Closed | |  |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| If concrete classes Range and TimeData are removed from the API (see linked items) this leaves the "util" package only with SystemOfUnits.  That alone is little justification for a whole package, and given the (also optional, see Liblets or ServiceLoader in platforms like ME being an optional MEEP feature!) dependency to SystemOfUnits from the SystemOfUnitsService, we could better move it to the top level package javax.measure or a similar (mandatory) one.  In very small environments a full scale unit system like SI or Imperial would be an overkill, but implementing a minimal custom specific unit system that only exposes one or a few units used in these devices sounds acceptable to be JSR compatible. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 08/Aug/14 ] |
| I agree on the rationale of removing this package, if we take out the other classes! |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-32] [Remove functional interfaces that are out of JSR-363 scope](https://java.net/jira/browse/UNITSOFMEASUREMENT-32) Created: 05/Aug/14  Updated: 26/Aug/14  Resolved: 25/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Minor |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | jdk8 | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-31~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-31) | Remove javax.measure.util.TimedData | Resolved | | depends on | [~~UNITSOFMEASUREMENT-30~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-30) | Remove javax.measure.util.Range | Closed | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-8~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-8) | Could remove BiFactory from API | Resolved | | is related to | [~~UNITSOFMEASUREMENT-49~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-49) | Specialize ConversionOperator | Resolved | | is related to | [~~UNITSOFMEASUREMENT-50~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-50) | Discuss \*Supplier functional interfaces | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-46](https://java.net/jira/browse/UNITSOFMEASUREMENT-46) | [Remove MinimumSupplier and MaximumSup...](https://java.net/jira/browse/UNITSOFMEASUREMENT-46) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-47](https://java.net/jira/browse/UNITSOFMEASUREMENT-47) | [Merge Converter and UnitConverter](https://java.net/jira/browse/UNITSOFMEASUREMENT-47) | Sub-task | Resolved | keilw | |  |
| **Tags:** | design, java8, portability |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The following interfaces are not specific to unit of measurement. They are covering a wider scope (e.g. they could be defined in java.util.function), in which case I think that JSR-363 should not take the responsibility to define them alone (unless those interfaces are really needed for Unit of Measurement, but I do not think that this is the case):   * javax.measure.util.MinimumSupplier (was used only by Range, which I suggest to remove). * javax.measure.util.MaximumSupplier (was used only by Range, which I suggest to remove). * javax.measure.function.Nameable - wider scope than JSR-363. * javax.measure.function.ValueSupplier - duplicates java.util.function.Supplier. * javax.measure.function.Converter - many major projects define their own ObjectConverter interface (e.g. Spring, Apache Common Converters, Camel, Jackrabit, etc.). I that JSR-363 should avoid adding an other converter in the mix. * BIFactory - I think it would rather be the scope of other JSR like dependency injection or inversion of control framework. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 05/Aug/14 ] |
| Seems fine for MinimumSupplier and MaximumSupplier, see the other tickets. For all others Lambda requirements for the API as opposed to RI or downstream applications must be analyzed.  All of these interfaces are designed to work with Lambda expressions in a Java SE8+ environment while strictly avoiding direct reference to e.g. the mentioned java.util.function.Supplier (which won't work in ME 8 or prior SE versions) So mimicking the Functional signature of Supplier while not directly importing it or @FunctionalInterface is quite deliberate and on purpose. Unless it's totally outruled, the API and implementations benefit from them, we should keep those. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 13/Aug/14 ] |
| I believe the best strategy is remove these functions and do the similar as money: Create a class with predicates and useful things to lambda and Stream: <https://github.com/JavaMoney/jsr354-ri/pull/35> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Aug/14 ] |
| Not sure, if that's right for all of them. We'll have to explore the Money solution and try something similar here.  @Otavio, you suggested an abstraction to QuantityFactory in the API, do you now feel, it should only exist in implementations? (that isn't just about Lambdas btw, more about making this available to the API or not) So aside from the name BiFactory is exactly about that. We may call it QuantityFactory or something else, then a certain level of generic would be useless/annoying and better be dropped. In any case, I'd rather reopen <https://java.net/jira/browse/UNITSOFMEASUREMENT-8> for that.  ValueSupplier is worth looking at, e.g. now that Range/TimedData are in RI already, it is only used by Quantity. To really use the Java SE 8 supplier, an implementing type cannot call such method getValue(), it'll have to be get() only with a matching return type.  I see value in the Nameable interface as it is used at least across the API by Unit and SystemOfUnits already. While JSR 354 prefers to keep the currency name language and locale-dependent, there are plenty of Unit catalogs like UCUM where a globally unique (English or otherwise) name of a unit is defined that won't change even if you translate or format it in another locale. Thus, getName() for the unit makes sense, and it also does for SUO. Allowing a lambda-capable access to that name is the purpose of this functional interface, it has nothing to do with @Named in JSR 330, etc. thus it makes sense within the API like e.g. that limited ValueRange does to java.time  We may see if Converter and UnitConverter could better be merged into one, the purpose and function of the actual converter seems out of the question. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 13/Aug/14 ] |
| Yes, you are right. How API, the JSR 363-API cannot impose which measure the implementations can have. If this factory goes to impl, I could create specific measure and following the specifications, so standardization to measure unit #JSRWIN. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 13/Aug/14 ] |
| About Value Supplier and other interfaces, I understand, but maybe create interface is interesting in impl-se because lambda, maybe you can think in another way to wrapper it just in SE plataform. So if I just use it in SE plataform, I could use Supplier present in SE. <http://docs.oracle.com/javase/8/docs/api/java/util/function/Supplier.html>  UnitSupplier: could removed the same reason. Nameable: ok UnitConverter and Converter could be merge on one or UnitConverter be specialized of Converter |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Aug/14 ] |
| Unfortunately Supplier has its own method signature, and for compatibility reasons I don't think we'd be happy to remove the value from the API, but an implementing class may add a get(), implement Supplier and define what it wants to return (that could be either the value, unit or both, or even the whole Measurement/Quantity instead)  A large number of converter implementations require a getValue() method, too, e.g. for conversion factors (RationalConverter, etc.) but those are RI specific, so in theory one could use something else, but the signature between RI and SE implementation should ideally match, otherwise we risk fragmentation. Converter has not been a single method interface yet, so merging it with UnitConverter seems OK. The other ConversionOperator offering a single to() method is however useful and it meets every aspect of the FunctionalInterface definition in SE 8. There is a fine line between those things e.g. defined by JavaFX, but just look at the duplication between JavaFX and JSR 310 with entire base types like Duration being duplicated, doing exactly the same and in an ideal world, they could be reused, in reality even Java 9 or 10 may not see that, especially because 310 is too monolithic and almost impossible to modularize like we try better here... |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 14/Aug/14 ] |
| I do not think that we need to provides any functional interface for Lambda support. The functional interfaces are needed by API that **consume** the information supplied (e.g. the JDK 8 stream API, for example Stream.generate(Supplier<T> s)). There is no such API in JSR-363. There is no need for Unit API to implement any functional interface for allowing their use. For example if someone wants to get a stream of unit names from a collection of Unit<?>, he can write in JDK8:  Set<Unit<?>> units = ...;  Stream<String> names = units.stream().map(Unit::getName);  As we can see, there is no need for a Nameable interface.  I think we should remove all javax.measure.function interfaces except UnitConverter. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Aug/14 ] |
| That is not correct, EVERY interface with a single method there is a Functional Interface, see <http://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html#package.description>  It's not about the @FunctionalInterface annotation, but Java 8 (if above is correct and not a mistake in the JavaDoc? ) will recognize the pattern in Java 8. If this only works for JDK 8 functional interfaces for some reason, then Lambdas would be flawed. getName() is used by both the Unit and SystemOfUnits.  getValue() currently by every single converter in RI or SE implementation, and of course ConversionOperator provides a functional method used by Measurement (since that has a UnitSupplier, potentially the only interface used by no more than Measurement) Parser (in the "format" package) is used by multiple formatters in a consistent way that also feels similar to JSR 354 (I plan to create a ticket for further harmonization of one format method with 354, but that's not directly related to this ticket)  Will need more investigation, I don't think the 2 lines above come from a larger executable demo, do they?  There are explanations like  // Method invocation context stream.filter(e -> e.getSize() > 10)... or // Cast context stream.map((ToIntFunction) e -> e.getSize()) in the package description for Functional Interfaces. None clear enough, if getSize() must be defined in a separate interface for this to work. If it does, then the likes of ValueSupplier are inevitable for Lambda calls like  stream.filter(q -> q.getValue() > 10)... If it simply worked on any interface or class with multiple getters that do not implement such functional interfaces, then we may be able to reduce or merge those interfaces. Proof required. According to JavaDoc I would assume these interfaces are necessary. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 14/Aug/14 ] |
| My previous comment is unrelated to the @FunctionalInterface annotation. The comment was saying that javax.measure.function is not needed for Lambda, no matter the annotation. Can you provide an example of a lambda expression that we can not write without Nameable for instance? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Aug/14 ] |
| Otavio said he did something with 354. Will have a look at it when I can. That should enable us to understand what could be removed or not. Of course the API of 354 is tightly coupled with Java SE 8, so it may refer to java.util.function.Supplier or use @FunctionalInterface both a no-go here as we want to ensure a relatively wide compatibility of the API, especially ME. So we don't want to put Supplier into the API, but certain other interfaces that are recognized by Java SE 8 or above could be necessary. At least that's what SE 8 JavaDoc tells us. Hackathons, no later than JavaOne could also help clarify that, unless we get such PoC for Lambda earlier. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 14/Aug/14 ] |
| We do not need to implement a JDK8 interface ourself, and we do not need our own interfaces neither, as demonstrated by the units.stream().map(Unit::getName) example. For those who are more familiar with anonymous classes, the above Lambda expression is equivalent to:  units.stream().map(new Function<Unit<?>, String>() {  @Override  public String apply(Unit<?> unit) {  return unit.getName();  }  });  As we can see, the implementation of the JDK8 Function interface is provided by the user, not by the JSR-363 library. If the user is not using JDK8, he will implement an other interface - JSR-363 does not need to know which one. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Aug/14 ] |
| Did you also try that with an OLD Unit from the 0.6 legacy API? (e.g. Opower's patch of JScience would be sufficient to test)  If a user implemented Supplier (in a Java SE 8 variation) they won't be able to use the Unit or Quantity type in their application code but only an implementation type like MyQuantity. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 14/Aug/14 ] |
| Yes, I tried with JSR-275. I don't see anything that would prevent a user from using Unit or Quantity in Lambda expressions - we do not need to define any functional interface for that. Even if we define functional interfaces like Nameable, as far as I can see those interfaces would be useless for someone who wants to use the JDK8 API (or any other library which is not aware of javax.measure.function interfaces). |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/Aug/14 ] |
| Can we put code snippets for that somewhere, at least attachment here in JIRA? In fact, functional interfaces are more beneficial to users of Java SE 8, in others (like ME 8) it saves code and unifies behavior but they would not be able to use Lambda. If someone wanted to implement the JDK Supplier method, then they'd have both, but that doesn't take the functional interfaces in javax.measure away from them if they use the JSR specific API, We can't use things like java.time in the API either without a lock-in making the API useless to ME and similar environments.  I see, JavaMoney keept its CurrencySupplier, etc. so @Otavio, will you remove those or is there a reason to keep them there? |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 15/Aug/14 ] |
| I believe it isn't necessary do anything that contains in Java SE 8, I Believe the best strategy is make an easy API to encourage to use lambda and have the code more readable. On money we created: <https://github.com/JavaMoney/jsr354-ri/blob/master/src/main/java/org/javamoney/moneta/function/MonetaryFunctions.java>  You can see the examples of use in this path: <https://github.com/JavaMoney/jsr354-ri/tree/master/src/test/java/org/javamoney/moneta/function>  I hope this weekend do an article about it and the funcionalities.  I am too agree that funcionalities is useful to SE user, so I believe the best strategy is be with SE impl.  About CurrencySupplier, I don't see any benefits of this class instead of use Supplier<Currency>. <http://docs.oracle.com/javase/8/docs/api/java/util/function/Supplier.html> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/Aug/14 ] |
| About CurrencySupplier, then you may like to remove it from 354? Of course the method signature can't be applied twice, so Supplier<CurrencyUnit> and Supplier<NumberValue> to replace both will fail (I tried that before ) A major difference between 363 and 354 is that we are not tied into SE 8, so Supplier<Unit> or Suppplier<Number> is not available at all. Not to forget the same problem in Unit where you cannot implement Supplier twice not even in subclasses.  We must not stick to Lambda for unit conversions, that's for sure. There is no lambda in ME 8, so unless we completely eliminated every single conversion (and left it to implementations to do whatever they want) it must remain on an API level in ways that don't require Lambda or Java SE 8. Thus an equivalent to MonetaryFunctions could be a nice add-on for things like the Range, Min/Max/Average, but these are out of scope beyond the simple API level operations defined in the Quantity interface.  If you can demonstrate how you'd use the JDK 8 built-in Supplier more than once in an implementing class, we could think of moving some of the suppliers, but defining them redundantly in the SE implementation (e.g. JSR 354 currently has an unnecessary getCurrency() method that's already in CurrencySupplier, either the functional interface could go or the method in MonetaryAmount wasn't necessary) would be a waste. So if an interface with half a dozen getXX() methods works fine with any Lambda need we could imagine right now, then merging a few of them like UnitTransformer or \*Supplier into one interface could work.  As soon as you must have them in an SE implementation, there is no reason to move them there, because the information (getName(), getSymbol(),...) must be in the interface anyway, so something like BaseUnit implements Unit, Supplier<String>, Nameable, SymbolSupplier,... in the SE implementation would not make any sense at all. Each of the functional interfaces do no harm in the API should they need to be in separate interfaces with one method each (see the Supplier dilemma, that interface is just a default example after all, it can't be applied more than once per inheritance chain)  Can you (will also look at it on the weekend) provide actual code examples, not just a blog post to analyze where we need functional interfaces and where a signature with dozens of different getters or other methods was fine?  Thanks, Werner |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 15/Aug/14 ] |
| Ok, I will do that. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 21/Aug/14 ] |
| Hi first sorry by delay. I wrote an article about it. And I believe in JSR 363, We can create functions that make usefully the Stream's use. Functions to sort, mapper, reducer, filter. But I believe that can be done when all API is done. BTW, IMHO the best strategy is do to me when is finished fork this project and focus to SE with these lambdas functions. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 21/Aug/14 ] |
| FORK? Sorry, that IMHO is a terrible idea, see the mess around 310 and other APIs baking in SE 8. Implementing libraries, sure, we can have as many platform-optimized apps or libraries as we feel are appropriate, but there must be only one "javax.measure" API.  The "SE Implementation" is a fork of the RI already. That makes perfect sense. Whether some interfaces would be better used in such SE specific implementation, we'll find out. Oddly some of the most perfect Lambda examples look like this: <http://winterbe.com/posts/2014/03/16/java-8-tutorial/> Note he has a:  @FunctionalInterface interface Converter<F, T>  { T convert(F from); }  Which reassembles some of the functional interfaces in the API quite a bit (no matter if the method is called to(), convert() or convertTo())  If his further example: Converter<String, Integer> converter = (from) -> Integer.valueOf(from); Integer converted = converter.convert("123"); System.out.println(converted); // 123  was to work with API level code defined by a Spec, then such interfaces regardless of the particular name or package should be in the API. Cou can call the first line in a traditional non-Lambda way if you are under SE 7 or ME 8, so no reason to criple the spec and API.  If it is SE or implementation specific, then you end up with something like SEConverter as opposed to RIConverter or not have any converter in the RI at all. Something that converts units or other essential things does seem best in the API. Other "helpful stuff" or special functions like Min/Max/Average, or similar (like in JavaMoney) probably can be implentation specific.  Looking forward to your examples or essay. And at least in the JavaOne "sprint" we can look more closely at this. It allows input by community members e.g. during a Hackergarten, too |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 22/Aug/14 ] |
| @keilw you are right. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 24/Aug/14 ] |
| I'm a little bit lost in the flux of comments. Can we check point-by-point where we agree and where we disagree?   1. We need UnitConverter - this task is in the core of JSR-363 purpose. 2. BiFactory, ConversionOperator, Nameable, ValueSupplier are not directly related to JSR-363. They are of wider scope. 3. BiFactory, ConversionOperator, Nameable, UnitSupplier, UnitTransformer and ValueSupplier all have equivalent interfaces in java.util.function. Those javax.measure.function interfaces do not add any new functionality. 4. **We do not need to implement any interface for allowing lambda usage**. See my example in a previous comment. 5. As a consequence of the above, we do not need to introduce a dependency to JDK 8, and we do not need to define our own interfaces (other than UnitConverter) neither. 6. Admittedly, implementing javax.measure.function-like interfaces may save a few lines of code on platforms that do not support lambda expressions. But developers can achieve the same result with anonymous classes. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Aug/14 ] |
| That's why I created sub-tasks  BiFactory is related to Otavio's aim of creating a more general interface for QuantityFactory. It may be generic in name, but without completely renaming each classes in RI, SE or other implementations the purpose exists. If we ended up naming the interface QuantityFactory, it might end up with less generics.  As mentioned, Supplier in JDK 8 neither gives us getters for Unit nor Name or Value. Nameable generalizes both Unit and SystemOfUnits. It would lead to unnecessary code-duplication if we put it into each of them.  None of the 3 Supplier-like interfaces can ever be replaced by Java 8 Supplier (as it only has a single get() method that works exactly ONCE per class) but in theory they could be merged into core types (Unit or Quantity) with less impact.  I created an SE 8 console demo with a few simple Lambda calls, including Unit and Stream, etc. Some getXX() methods like getSymbol() or getName() seem to work, thus what's said above.  However in the FunctionDemo: ConversionOperator<String, Integer> converter = (from) -> Integer.valueOf(from); Integer converted = converter.to("123"); System.out.println(converted); // 123 works (regardless that we convert only a string to an integer here) While  //Quantity<Length> conv2 = (from) -> Integer.valueOf(from); (assuming the functional interface was destroyed)  fails with the compiler telling "the target type of this expression must be a functional interface". As a matter of fact, the way UnitConverter contains multiple convert() methods, it also does not serve as "functional interface", but we have so few types that only deal with Unit that it makes little sense to put it into a separate package. Cluttering the top level one also seems a bit odd, though in some cases like SystemOfUnits that was done before, so in a decent way the root package may be OK with one or the other type (just not 30 like with JavaMoney if we can  About using anonymous code, in environments where Lambdas are not available that is a legitimate alternative, but Oracle made it clear in recent JSR proposals like CDI 2 or upcoming MVC, etc. that they insist on Java 8 support where possible. So offering Lambda-capable code without tying it into SE 8 (because this JSR must scale from ME 8 to SE/EE) is a good signal and does not do any harm to ME or Java <=7 users.  We planned to look at those e.g. at a F2F during JavaOne. For some sub-tasks like Quantity/BiFactory Otavio said he's looking into it, too. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 24/Aug/14 ] |
| We do not need javax.measure.function interfaces for lambda support even if java.util.function.Supplier has a single get() method. If someone wants to work on unit name, he can write:  Stream<String> names = units.stream().map(Unit::getName);  If he wants to work on unit symbol:  Stream<String> names = units.stream().map(Unit::getSymbol);  And so on for all compatible methods, no matter if named getUnit() or whatever. Why would we need custom interfaces? Repeating my previous comment, **we do not need to implement any interface for enabling lambda support**. So the fact that Supplier has only one get() method is not a problem. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 24/Aug/14 ] |
| Got the point for Quantity factory. But then, I think that we should provide a QuantityFactory interface or something similar, rather than trying to be more generic. If someone needs a more generic interface, I think that the chances that he would use the JSR-363 interface is rather low. The problem would be similar to ConversionOperator - there is many projects providing similar interfaces, and many of those projects are much more widespread than JSR-363 (for example Spring). Adding a JSR-363 interfaces in the mix may only increases the chaos, since I think there is few chances that the JSR-363 interface would be preferred to the Spring (or other projects) ones. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 24/Aug/14 ] |
| We could potentially give it a more specific name in some cases. See JSR 310 it used a totally generic (except specific to Long, but otherwise no Date relation whatsoever ValueRange like the one we moved to implementations as of now (of course there is no distinction between Spec, API and RI in 310, so it's just THERE ;-D)  So in a few cases where JDK does not provide anything we may use, defining a slightly more generic interface is not forbidden. Both should ideally have a "functional" signature, but where arguments are less generic, see UnitTransformer (specific to UnitConverter, but with a functional signature) |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 25/Aug/14 ] |
| The JSR-310 java.time.temporal.ValueRange class seems specific to the time domain, as the package name suggests. It has methods expecting TemporalField argument, and methods that are usually not found in general-purpose ranges like getLargestMinimum() and getSmallestMaximum().  On javax.measure.function.BiFactory interface, it seems to me that java.util.function.BiFunction provides similar functionality? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| Well we could merge that with QuantityFactory, but ideally the interfaces should not lose the Functional nature, otherwise they are useless with Lambdas. Most of the huge, bloated types of JSR 310 with all the toMilli(), ofNanoAsLong() etc. methods never work with lambdas. AFAIK only TemporalQuery and TemporalAdjuster are true functional interfaces in 310 each also annotated.  The likes of UnitTransformer, ConversionOperator, BiFactory or Parser all work like that. Parser IMHO must be fairly generic, there are several usages, each Formatter for Unit, Measurement, etc. but also custom parsers like UCUM use that. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 25/Aug/14 ] |
| I agree with @keilw, interfaces should be used to make the code better and more readable. About lambda, we can create some functions to make the Stream and lambda experience easier, but Just in SE implementation. Example using Money with Stream provides of API: <https://github.com/JavaMoney/javamoney-examples/tree/master/functional-example> |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 25/Aug/14 ] |
| The examples from Java Money use lambda with libraries designed for working with Java Money interfaces. I mean, they have methods that **expect in argument** an instance of a Java Money interfaces for doing their work. This is not the same than having classes **implementing** those interfaces.  In JSR-363, we do not have any function working with javax.measure.function interfaces other than UnitConversion. So I don't think that the Java Money examples apply. Again, does anyone can provide a single use case showing the need for javax.measure.function interfaces (except UnitConversion) with the current JSR-363 API? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| Actually none of above JavaMoney demos contain much of "Lambda" functionality like "->" or "::".  A somewhat comparable example to unit conversion would be: MonetaryAmount money = Money.of(10, real); System.out.println(money.with(ecbDollarConvertion)); System.out.println(money.with(imfDollarConvertion));  But I don't see too much of a similarity, actually the closest I can imagine would be  transform(UnitConverter converter) defined in a "functional" UnitTransformer, to call it like shown above, it would probably work to merge that method signature into Unit, therefore losing the FunctionalInterface, thus demos like I drafted here: <https://github.com/unitsofmeasurement/unit-demos/tree/master/console/se/src/main/java/tec/uom/demo/se/lambda>  wouldn't work. MonetaryAmount extends 3 functional interfaces, CurrencySupplier, NumberSupplier and (in Java 8 that also works like it now) Comparable but other than that is isn't a functional interface by itself. Hence the with(MonetaryOperator) method is Lambda-inebt.  See JSR 310, there the TemporalAdjuster is one of the few true functional interfaces. Would MonetaryAmount require something similar for the with() method, then it had to extend another functional interface like MonetarySOANDSO (not sure what to call it, "Adjuster" was probably suggested but the with() method doesn't seem right then ) that contains only a single with() method.  As long as JSR 354 does not eliminate all "\*Supplier" interfaces, there seems room for that at least for value or Unit here, too. Nameable is a bit more generic, while UnitSupplier is an integral part of what the JSR does and specifies, it's a functional interface that provides a Unit hence I see more justification in that (like CurrencySupplier in 354) than maybe a generalized getName() method. So   * UnitTransformer * UnitSupplier * \*Factory (could be QuantityFactory or something else) * ConversionOperator (if it was to become less generic, then the argument for to() might be a Unit, but then the name must be distinguishable form UnitConverter ) all make great sense in the scope of a Unit-API. The others have room for pruning.   A Hackergarten session is planned at JavaOne with LJC and others around JSR 354, I proposed we also asked Heather for a similar 363 session, then we can look at what works with the Lambda and Java 8 "geeks" and what might be factored out. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 25/Aug/14 ] |
| @desruisseaux Both are measure, so they have measure unit and the value. In money there is the currency unit, that can be dollar, euro, etc. and the value that is a number In measure have the unit as day, minutes, hours, etc. and the value that is a number.  @keilw Yes, the API uses lambda it's just hide in implementations, you can see in the source. <https://github.com/JavaMoney/jsr354-ri/blob/master/src/main/java/org/javamoney/moneta/function/MonetaryFunctions.java> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| We moved getValue() usually to be number lower to the Quantity interface as of now, but getUnit() (from UnitSupplier) sits on top. As of now Measurement takes a second generic argument, probably something to tweak, if we make the assumption, Quantity will never return a value other than a Number subclass. Money (for some higher precision BigDecimal mimicking) adds a wrapper interface called NumberValue which speaking very conservative would be out of scope for JSR 354, too It brushes out some shortcomings of Java decimal support which hopefully are not of such importance here.  I know, that's on a RI level. Which is why I am not sure, if CurrencySupplier is essential to JavaMoney, see: MonetaryAmount::getCurrency  If all of these work without referring to CurrencySupplier then it could be worth considering to merge that into MonetaryAmount and where applicable raise a question about the likes of UnitSupplier, etc. but not the conversion or transformation related ones.  JSR 310 contains a bad mish-mash of API and RI, but everything in java.time.temporal is pretty much the only API it has, thus TemporalAdjuster etc. are functional interfaces on an API level. We must not directly use 310 outside SE 8 implementations, but we shall try to learn from where it applies functional interfaces, to some extent also JSR 354, which IMHO also has room for discussion, see CurrencySupplier.  @FunctionalInterface annotation is just syntactic sugar for JavaDoc, the compiler doesn't care, as the SE 8 demos for 363 show. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 25/Aug/14 ] |
| @otaviojava Thanks for the link to MonetaryFunctions. I had a look, but what I see in this class does not seem to need functional interfaces in the JSR-354 API. For example:  public static Collector<MonetaryAmount, ?, Map<CurrencyUnit, List<MonetaryAmount>>> groupByCurrencyUnit() {  return Collectors.groupingBy(MonetaryAmount::getCurrency);  }  would work no matter if MonetaryAmount is a functional interface or not. So I would like to ask again: can someone provides a code demonstrating the need for javax.measure.function interfaces? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| See <https://github.com/unitsofmeasurement/unit-demos/tree/master/console/se/src/main/java/tec/uom/demo/se/lambda> |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 25/Aug/14 ] |
| I had a look at those demos. StreamDemo does not need the javax.measure.function interfaces. And FunctionalDemo is weird: the following line makes a unnecessary usage of lambda (the from argument is never used):  ConversionOperator<Unit<Length>, Quantity<Length>> converter2 = (from) -> AbstractQuantity.of(10, SI.METRE);  That line could be:  ConversionOperator<Unit<Length>, Quantity<Length>> converter2 = AbstractQuantity.of(10, SI.METRE);  or simply:  Quantity<Length> converter2 = AbstractQuantity.of(10, SI.METRE); |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| Please discuss ConversionOperator further in <https://java.net/jira/browse/UNITSOFMEASUREMENT-49> to unclutter this parent task  Same for \*Supplier interfaces under: <https://java.net/jira/browse/UNITSOFMEASUREMENT-50> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 25/Aug/14 ] |
| See sub-tasks for easier handling of individual items. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-31] [Remove javax.measure.util.TimedData](https://java.net/jira/browse/UNITSOFMEASUREMENT-31) Created: 05/Aug/14  Updated: 10/Aug/14  Resolved: 10/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 1 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-33~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-33) | Remove javax.measure.util Package | Resolved | | blocks | [~~UNITSOFMEASUREMENT-32~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-32) | Remove functional interfaces that are... | Resolved | |  |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The javax.measure.util.TimedData class is defined as a (name, value, timestamp) tupple. This construct seems unrelated to a unit API. It does not have an unit and is not used anywhere in the API. Some other issues are:   * The problem that TimedData tries to solve is not clearly identified (use case?) * The unit and origin of the timestamp is not specified. I presume that this is milliseconds since January 1st, 1970, but the Javadoc does not said anything about that. * Some may argue that the timestamp should be a java.util.time object instead.   But most important, TimedData seems out of JSR-363 scope. I think this object should be left to dedicated libraries like JScience or other. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 08/Aug/14 ] |
| While I understand that it may be out of scope of this JSR, everywhere we use a Unit, we normally tie in a timestamp. That's because every time we capture sensor data and log it, we need it on time, so many analysis are done.  So, in every firmware we write, we have an object that has a sensor id, a timestamp and a value. This class would be useful for us, but I agree it seems out of scope.  Lastly, remember this is tied to CLDC 8, so use of java.util.time is not an option. The long timestamp there has the same meaning as System.currentTimeMillis(), what's normally available in any Java ME environment.  Regards, Leonardo. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Aug/14 ] |
| There are examples like OSGi Measurement: <http://www.osgi.org/javadoc/r4v42/org/osgi/util/measurement/Measurement.html> with a long based timestamp, too.  If you feel that aside from a more "Time Series" approach the TimedData class tried to cover, this was a requirement on the API level, please raise a separate ticket.  Regards, Werner |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 10/Aug/14 ] |
| Hello Leonardo  While I understand that timestamp is important in many scenarios, I think that given that JSR-363 does not need timestamp for itself (I mean, there is no place in the JSR-363 API where we need to express a TimedData), we can leave time series structures to other libraries for now. Time series is an other vast topic, and I think it is safer to watch for the experience gained by other projects first. It would be trivial to add a TimedData in a future version if peoples wish, since it does not have any impact on the rest of JSR-363. But it would be very difficult to fix TimedData if, after standardization, someone tries to design a larger time-series framework and discover that TimedData does not fit well in the bigger "time series" picture.  Regards, Martin |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Aug/14 ] |
| Thanks for your input. I moved it recently. Into both RI and the SE equivalent. The latter allows a few extra things like referring to Functional Interfaces where appropriate or also express the timestamp in Java 8 Instant. While RI sticks to long as a rather common form of holding timestamps, see OSGi or Ford's OpenXC.  Regards, Werner |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-30] [Remove javax.measure.util.Range](https://java.net/jira/browse/UNITSOFMEASUREMENT-30) Created: 05/Aug/14  Updated: 08/Aug/14  Resolved: 08/Aug/14 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Bug | **Priority:** | Major |
| **Reporter:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 2 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-33~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-33) | Remove javax.measure.util Package | Resolved | | blocks | [~~UNITSOFMEASUREMENT-32~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-32) | Remove functional interfaces that are... | Resolved | |  |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The javax.measure.util.Range class as it stands now seems unrelated to unit of measurement. It contains a minimum and maximum value, without unit, and is not used anywhere in the Unit API. Some issues with Range are:   * It is out of JSR-363 scope. * It is one way to handle range, not the only way. For example for some peoples, the information about whether the minimum and maximum values are inclusive or exclusive is very important. * Some peoples may argue that the resolution parameter (or step, or increment) is not part of a range. * Different peoples will want a different set of operations (union, intersection, etc.) on ranges.   I suggest to leave Range to other libraries like JScience or any other. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-29] [Do we plan to change Unit Testing Framework?](https://java.net/jira/browse/UNITSOFMEASUREMENT-29) Created: 03/Aug/14  Updated: 20/Aug/14  Resolved: 20/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | question, test-framework |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Is there a benefit in migrating tests to TestNG like e.g. JavaMoney did? If desired to migrate, some tests may have to be rewritten --> action item |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 08/Aug/14 ] |
| As I mentioned during our call, I don't see any clear advantage for this switching. So I'd close as "won't fix". |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Aug/14 ] |
| See comment, this has been open for 3 weeks after Leo hinted he doesn't see a benefit for the project now in doing so. I second his opinion and we won't do this for now. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-28] [Epic for infrastructure tasks](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) Created: 01/Aug/14  Updated: 22/Oct/16  Resolved: 22/Oct/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.x](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17750) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Epic | **Priority:** | Trivial |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Name:** | Infrastructure |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-27] [Epic for design related tasks and issues](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) Created: 01/Aug/14  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.0](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16917) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Epic | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Name:** | Design |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-26] [Set up Jenkins job on GeoAPI](https://java.net/jira/browse/UNITSOFMEASUREMENT-26) Created: 01/Aug/14  Updated: 15/Aug/14  Due: 05/Aug/14  Resolved: 08/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-41~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-41) | Make API available to Snapshot Maven ... | Resolved | |  |
| **Tags:** | build, maven, server |
| **Epic Link:** | [Infrastructure](https://java.net/jira/browse/UNITSOFMEASUREMENT-28) |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Unit-API 0.6 has a Jenkins instance provided by GeoAPI: <http://jenkins.geotoolkit.org/job/Unit-API/>  A similar Jenkins job for JSR 363 API (if the original name Unit-API is best untouched, then in Jenkins that could be "JSR363-API" or similar) and sooner or later also for the RI (JSR363-RI or similar) is desired. Whether a CI build of the TCK is also useful, we shall see later, but for now API and RI should be built there.  Similar to www.unitsofmeasurement.org either on java.net or (if that was easiest) in a matching subdirectory of the www.unitsofmeasurement.org site output of "Maven Site" and any Asciidoc that can be properly generated shall be made available. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 03/Aug/14 ] |
| I do not know how to deploy on java.net. If there is a link about that somewhere, I could take a look. Otherwise the easiest would be to deploy on a www.unitsofmeasurement.org subdirectory. For example what about the following for the Javadoc?  www.unitsofmeasurement.org/jsr-363/apidocs  I can generate javadoc easily, however attempts to generate the Maven web site currently fails with java.lang.NoClassDefFoundError: org/sonatype/aether/graph/DependencyFilter. Do we really need the Maven web site, or is the javadoc sufficient? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Aug/14 ] |
| I guess JavaDoc seems fine for now.  The GitHub site (though it is not under www.unitsofmeasurement.org, but that is a minor detail, we could synchronize the content coming out of JBake at some point) has a wiki-like nature, so except a few pages like Team, etc. Maven Site is not that urgent.  I just did a rework of the Spec document, so the Asciidoc file is outdated. Will try to convert it, but for now a PDF (manually uploaded to java.net) export may be best. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 03/Aug/14 ] |
| P.s.: If you start working on it, feel free to "start progress" in JIRA. Only the assignee can do that.  Saw the progress, thanks. Similar to downstream jobs like "Geo-API" I assume something similar at least for "JSR-363-RI" can be easily added soon, too? Unless you plan to do this now, we can also define another task later. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 05/Aug/14 ] |
| * Jenkins job: <http://jenkins.geotoolkit.org/job/JSR-363/> * Maven site: <http://www.unitsofmeasurement.org/jsr-363/index.html> * Javadoc: <http://www.unitsofmeasurement.org/jsr-363/apidocs/index.html>   The reference implementation does not yet have a Jenkins job. First problem to solve: do we really needs unit-ri to have unitsofmeasurement-parent as a parent? Because if so this is yet an other job that we need to setup on Jenkins in order to be able to build unit-ri. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 08/Aug/14 ] |
| Thanks for the update! I updated the java.net page with the links pointing to this deployments. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Aug/14 ] |
| Seems OK, please open follow-up if there's something missing, e.g. similar jobs for RI, etc. |

|  |  |
| --- | --- |
| **[Create Spec Document](https://java.net/jira/browse/UNITSOFMEASUREMENT-2)** ([UNITSOFMEASUREMENT-2](https://java.net/jira/browse/UNITSOFMEASUREMENT-2)) [UNITSOFMEASUREMENT-25] [Adjust spec document to recent code changes](https://java.net/jira/browse/UNITSOFMEASUREMENT-25) Created: 01/Aug/14  Updated: 13/Dec/14  Resolved: 13/Dec/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Critical |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | document, specification |
| **Sprint:** | August, Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The initial document contains a few paragraphs or images that are still referring to earlier APIs (unitsofmeasurement.org 0.6 and/or JSR 275) so while the chapters look good, details have to be adjusted, removed or added.. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Dec/14 ] |
| Done for EDR |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-24] [Move arithmetic operations and getValue() from Measurement to Quantity](https://java.net/jira/browse/UNITSOFMEASUREMENT-24) Created: 01/Aug/14  Updated: 10/Aug/14  Resolved: 10/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | August |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The Measurement base interface contains arithmetic methods like add(), subtract(),... that make little sense other than with numeric values. Therefore it's best to move these to the Quantity interface.  While getUnit() to ensure unit type-safety is required in Measurement we also consider getValue() a better match for Quantity, allowing Measurement implementations to use different value representations, e.g. getMinimum()/getMaximum(),... |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-23] [Explore Stripped Implementations](https://java.net/jira/browse/UNITSOFMEASUREMENT-23) Created: 22/Jul/14  Updated: 15/Jan/15  Resolved: 15/Jan/15 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Trivial |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-96~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-96) | Clarify if the optionality of quantit... | Resolved | |  |
| **Tags:** | modules, portability |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This may not matter to the Final Release, depending on the pace Oracle makes pursuing it beyond striking it from Java 8. but in a recent JCP EC call with Mark Reinhold and Don Smith (as soon as the minutes are public, we'll update this ticket) they expressed Oracle's interest in "Sttripped Implementations" especially for SE and where it might make even more sense ME. The original post about dropping it is here, also including further details: <http://mail.openjdk.java.net/pipermail/java-se-8-spec-observers/2014-February/000064.html>  If Oracle makes this applicable to JSRs in the near future, we might very much look into that, otherwise it could be subject to an MR, hence the priority is lower, but it it worth discussing it where people are interested.  The Minutes of the July EC Call are now online, showing also a link to the proposal by Oracle: <https://jcp.org/aboutJava/communityprocess/ec-public/materials/2014-07-08/July-2014-Public-Minutes.html> |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/Jan/15 ] |
| After speaking with Mark Reinhold today, it seems less likely Oracle and the JCP will take the trouble of doing this for ME or SE 8. So beyond that Jigsaw Modularization is a future approach and the Optionality and Modularity already available and used by us as of ME 8 will have to to. In the unlikely event it gets revived, we may reopen this ticket later. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-22] [Documents (Spec, Documentation, Tutorials,...)](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) Created: 22/Jul/14  Updated: 22/Oct/16  Resolved: 22/Oct/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Epic | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Name:** | Documents |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-21] [Portability and Modularity](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) Created: 22/Jul/14  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Epic | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Epic Name:** | Portability |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-20] [Should we move Dimensionless into a (mandatory) core bundle?](https://java.net/jira/browse/UNITSOFMEASUREMENT-20) Created: 11/Jun/14  Updated: 15/May/15  Resolved: 15/May/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-100~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) | Define our criterion for inclusion / ... | Resolved | |  |
| **Tags:** | design, modules, portability, quantity, question |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| At the moment the Dimensionless type is part of the javax.measure.quantity package. This is not wrong, but if you assume the rather big quantity package optional, having a default type (sort of like Null or if you want Java 8 Optional) aside from Quantity itself in the base package could be good for modularity reasons. Shouldn't increase the base package too much, and for most SI units their quantities are well defined, see: <http://en.wikipedia.org/wiki/International_System_of_Units> only Radian and Steradian are Dimensionless. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 07/Apr/15 ] |
| If the minimal choice of API quantities (possibly just Dimensionless) was decided for, moving it e.g. to the SPI package sounds logical, otherwise it may best stay with other (base) quantities. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/May/15 ] |
| As we're defining a minimal set of quantities for the API, it seems Dimensionless is fine in the quantity package. If there was a different view, please reopen or create another task. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-19] [Adjust module structure](https://java.net/jira/browse/UNITSOFMEASUREMENT-19) Created: 08/Jun/14  Updated: 22/Jul/14  Resolved: 22/Jul/14 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-1~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-1) | Consider a separate unit-convert module | Resolved | |  |
| **Tags:** | modules, portability |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | June-July |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| While optionality and with it modularity are key requirements for this JSR (to meet different target devices and their memory constraints) it was discussed, whether we want a granularity similar to MEEP (with nearly a separate module/JAR for each package) or smaller, by combining different aspects like "format/parser/UI" and others like "Utility types" or an "SPI/Service" module into one or more.  Aside from that and the mandatory "core" module (with everything that has no further dependency on other types) the biggest component is the "quantity type system". Merging that with another bundle would be possible, but put a rather large burden on single-functional devices or e.g. parser/formatter implementations that are often type-agnostic beside base types like Unit or Quantity (e.g. UCUM) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Jun/14 ] |
| Adding or removing new modules is related to this task. Will consider closing the initial ticket allowing to discuss this further here. |

|  |  |
| --- | --- |
| **[Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)** ([UNITSOFMEASUREMENT-3](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)) [UNITSOFMEASUREMENT-18] [Factoring out SE only parts](https://java.net/jira/browse/UNITSOFMEASUREMENT-18) Created: 08/Jun/14  Updated: 06/Oct/14  Resolved: 06/Oct/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | implementation, java8, javase, portability |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| While RI components should be compatible with Java ME 8 those classes or other elements working only for Java SE should be moved into a separate SE implementation project (<https://github.com/unitsofmeasurement/unit-impl-se>) if not done so already and deleted from the RI. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 06/Oct/14 ] |
| UOM-SE created as separate project |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-17] [Should we support Measurement Lists?](https://java.net/jira/browse/UNITSOFMEASUREMENT-17) Created: 20/May/14  Updated: 05/Sep/14  Resolved: 05/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | New Feature | **Priority:** | Trivial |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, format, java8, parser, question |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| While it probably won't be more than an optional type in a module like "util" (similar to others like Range) how do you consider a "List Pattern" similar to Unicode CLDR: <http://cldr.unicode.org/index/downloads/cldr-24#TOC-Enhanced-units-structure> >A new <durationUnit> element specifies patterns for time durations involving combinations of hours, minutes, and seconds, such as "Video >length: 93:45". or > <listPattern> These are intended to be used for constructing formats such as "2 hours, 5 minutes", "2 hrs, 5 mins", or "2h 5m"  Essentially it is similar to TemporalAmount: <http://docs.oracle.com/javase/8/docs/api/java/time/temporal/TemporalAmount.html> in JSR 310, but unlike 310 there is no structural "array" of units in ICU4J.  Instead, e.g. MeasureFormat: <http://icu-project.org/apiref/icu4j/com/ibm/icu/text/MeasureFormat.html> allows formatting more than one measurement: MeasureFormat fmtEn = MeasureFormat.getInstance(ULocale.ENGLISH, FormatWidth.WIDE);  // Output: 1 inch, 2 feet fmtEn.formatMeasures( new Measure(1, MeasureUnit.INCH), new Measure(2, MeasureUnit.FOOT));  So it could be handled by Formatting/Parsing similar to ICU4J, where at most an array or collection of individual measurements is used, or via a structural type similar to JSR 310. I don't think we should overcomplicate the Measurement/Quantity type similar to TemporalAmount, but as with Range it could be provided in an optional module (or even just on RI level) if there's value to it. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 05/Sep/14 ] |
| I don't think we should support this in the API/RI. We could support this in an "extras" package.  This falls under the same category of TemporalAmount, surely. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 05/Sep/14 ] |
| It's still in the backlog, but thanks, that makes sense. JSR 310 unfortunately is quite a bad mix of "Spec/API" and RI as you probably know (and regardless of what "Joda Fanboys" may have voted for or not, that's why I feel either CLDC 8 or JSR 363 deserved a JCP Award more in that category ) and TemporalAmount could have declared a single unit like "MINUTES", etc. (the plural seems explained by the bad example given by Java Concurrency, while especially ICU4J does it better ) and a sort of List or "MultiAmount" would then hold them together. That is also how JSR 354 did it. We discussed a "Major/Minor" (for something like "10 Dollars and 5 cents") structure, but it was either dismissed or at least deferred into RI or some "Lib" module. We have the same for UOM, so we could think about it there.  I'll close this if you are happy and copy relevant content into <https://github.com/unitsofmeasurement/unit-lib/issues> Thanks, Werner |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 05/Sep/14 ] |
| Unless RI or other implementations were a better place, it is considered for UOM Libraries, see <https://github.com/unitsofmeasurement/unit-lib/issues/3> |

|  |  |
| --- | --- |
| **[Create Spec Document](https://java.net/jira/browse/UNITSOFMEASUREMENT-2)** ([UNITSOFMEASUREMENT-2](https://java.net/jira/browse/UNITSOFMEASUREMENT-2)) [UNITSOFMEASUREMENT-16] [Document Format](https://java.net/jira/browse/UNITSOFMEASUREMENT-16) Created: 14/May/14  Updated: 08/Aug/14  Resolved: 08/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | doc, question, spec |
| **Sprint:** | August, Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| For the Spec Document, these options and formats have been used by most JSRs in recent years:  An actual document file using Open Document Fomat (OpenOffice/LibreOffice) Collaborative Document like Google Docs Markup using Markdown, Maven Site or Asciidoc Especially Asciidoc has become extremely popular, used by nearly every JBoss JSR like CDI, BeanValidation, etc. and JSR 354 also introduced it very recently: <https://github.com/JavaMoney/jsr354-api/tree/master/src/main/asciidoc> |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/May/14 ] |
| Adding on-the fly PDF generation JSR 354 showed rather nicely, how Asciidoc can be used. Hope, we may be able to have a chat in London next week to clarify what format is best. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/May/14 ] |
| As of the London F2F, all participants found it a good idea to try Asciidoc. It worked for a number of JSRs like BeanValidation, CDI or more recently 354 (Money) and allows to keep API and Spec closer together including tagging/versioning both at important milestones. |

|  |  |
| --- | --- |
| **[Java 8 issues](https://java.net/jira/browse/UNITSOFMEASUREMENT-10)** ([UNITSOFMEASUREMENT-10](https://java.net/jira/browse/UNITSOFMEASUREMENT-10)) [UNITSOFMEASUREMENT-15] [RI JavaDoc](https://java.net/jira/browse/UNITSOFMEASUREMENT-15) Created: 14/May/14  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [1.0](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16917) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Won't Fix | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | java8, javadoc |
| **Sprint:** | Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| RI JavaDoc contains some entries that could cause issues with a more strict Java 8 JavaDoc:  71 warnings [WARNING] Javadoc Warnings [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractMeasurement.java:202: warning - Tag @link: can't find doubleValue(Unit) in javax.measure.Measurement [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractMeasurement.java:228: warning - Tag @link: can't find equals(javax.measure.Measurement, double, javax.measure.unit.Unit) in org.unitsofmeasurement.ri.AbstractMeasurement [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractMeasurement.java:348: warning - Tag @link: can't find valueOf(java.math.BigDecimal, javax.measure.unit.Unit) in org.unitsofmeasurement.ri.AbstractMeasurement [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractMeasurement.java:142: warning - Tag @link: can't find to(javax.measure.unit.Unit) in org.unitsofmeasurement.ri.AbstractMeasurement [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractMeasurement.java:348: warning - Tag @link: reference not found: javax.measure.unit.UnitFormat#getStandard standard [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractQuantity.java:201: warning - Tag @link: can't find doubleValue(Unit) in javax.measure.Measurement [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractQuantity.java:227: warning - Tag @link: can't find equals(javax.measure.Measurement, double, javax.measure.unit.Unit) in org.unitsofmeasurement.ri.AbstractQuantity [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractQuantity.java:347: warning - Tag @link: can't find valueOf(java.math.BigDecimal, javax.measure.unit.Unit) in org.unitsofmeasurement.ri.AbstractQuantity [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractQuantity.java:141: warning - Tag @link: can't find to(javax.measure.unit.Unit) in org.unitsofmeasurement.ri.AbstractQuantity [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractQuantity.java:347: warning - Tag @link: reference not found: javax.measure.unit.UnitFormat#getStandard standard [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:58: warning - Tag @link: reference not found: org.org.unitsofmeasurement.impl.system.SI SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:156: warning - Tag @link: reference not found: org.unitsofmeasurement.service.UnitFormatService [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:175: warning - Tag @link: reference not found: org.unitsofmeasurement.service.UnitFormat [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AlternateUnit.java:59: warning - @param argument "parent" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AlternateUnit.java:59: warning - Tag @link: can't find isSystemUnit() in org.unitsofmeasurement.ri.AbstractUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AnnotatedUnit.java:56: warning - Tag @return cannot be used in constructor documentation. It can only be used in the following types of documentation: method. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\BaseMeasurement.java:42: warning - Tag @see: reference not found: MeasureUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\BaseQuantity.java:42: warning - Tag @see: reference not found: MeasureUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\BaseUnit.java:43: warning - Tag @link: reference not found: org.org.unitsofmeasurement.impl.system.SI SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\TransformedUnit.java:79: warning - Tag @link: can't find isSystemUnit() in org.unitsofmeasurement.ri.AbstractUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:51: warning - Tag @link: can't find formatCompound in org.unitsofmeasurement.ri.format.MeasurementFormat [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:51: warning - Tag @link: reference not found: CompoundUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:200: warning - Tag @link: reference not found: CompoundUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:105: warning - Tag @link: can't find getStandardInstance() in javax.measure.format.UnitFormat [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:156: warning - @param argument "cursor" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:200: warning - Tag @link: reference not found: CompoundUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:200: warning - @param argument "value" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:200: warning - @param argument "unit" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:200: warning - @param argument "dest" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:51: warning - Tag @link: can't find formatCompound in org.unitsofmeasurement.ri.format.QuantityFormat [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:51: warning - Tag @link: reference not found: CompoundUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:200: warning - Tag @link: reference not found: CompoundUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:105: warning - Tag @link: can't find getStandardInstance() in javax.measure.format.UnitFormat [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:156: warning - @param argument "cursor" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:200: warning - Tag @link: reference not found: CompoundUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:200: warning - @param argument "value" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:200: warning - @param argument "unit" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:200: warning - @param argument "dest" is not a parameter name. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\DimensionalModel.java:68: warning - Tag @link: can't find getConverter in org.unitsofmeasurement.ri.model.DimensionalModel [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\DimensionalModel.java:82: warning - Tag @see: reference not found: LocalContext [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\QuantityDimension.java:49: warning - End Delimiter } missing for possible See Tag in comment string: "<p> This class represents a quantity dimension (dimension of a physical [WARNING] quantity).</p> [WARNING]  [WARNING] <p> The dimension associated to any given quantity are given by the [WARNING] OSGi published  {@link PhysicsDimensionService}  instances. [WARNING] For convenience, a static method {@link QuantityDimension#getInstance(Class) [WARNING] aggregating the results of all  {@link DimensionService}  instances [WARNING] is provided.<br/><br/> [WARNING] <code> [WARNING] QuantityDimension velocityDimension [WARNING] = QuantityDimension.of(Velocity.class); [WARNING] </code> [WARNING] </p>" [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\QuantityDimension.java:49: warning - Tag @link: reference not found: PhysicsDimensionService [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\QuantityDimension.java:110: warning - Tag @link: reference not found: DimensionService [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\QuantityDimension.java:110: warning - Tag @link: reference not found: DimensionService [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\CommonUnits.java:43: warning - Tag @link: can't find KILOMETRE in org.unitsofmeasurement.ri.util.CommonUnits [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\CommonUnits.java:43: warning - Tag @link: can't find HOUR in org.unitsofmeasurement.ri.util.CommonUnits [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\SI.java:47: warning - Tag @see: reference not found: SIPrefixOld [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\US.java:55: warning - @noextend is an unknown tag. [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\US.java:287: warning - Tag @see:illegal character: "58" in "http://en.wikipedia.org/wiki/Acre" [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\US.java:287: warning - Tag @see:illegal character: "47" in "http://en.wikipedia.org/wiki/Acre" [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\US.java:287: warning - Tag @see:illegal character: "47" in "http://en.wikipedia.org/wiki/Acre" [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\US.java:287: warning - Tag @see:illegal character: "47" in "http://en.wikipedia.org/wiki/Acre" [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\US.java:287: warning - Tag @see:illegal character: "47" in "http://en.wikipedia.org/wiki/Acre" [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\US.java:287: warning - Tag @see: reference not found: <http://en.wikipedia.org/wiki/Acre> [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:58: warning - Tag @link: reference not found: org.org.unitsofmeasurement.impl.system.SI SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\function\package-info.java:25: warning - Tag @link: reference not found: javax.measure.service.UnitformatService [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\package-info.java:134: warning - Tag @link: reference not found: org.unitsofmeasurement.quantity.Length length [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\package-info.java:134: warning - Tag @link: reference not found: org.unitsofmeasurement.quantity.Duration duration [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\package-info.java:134: warning - Tag @link: reference not found: javolution.context.LocalContext context-local [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\package-info.java:39: warning - Tag @link: reference not found: SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\package-info.java:39: warning - Tag @link: reference not found: US [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\package-info.java:39: warning - Tag @link: reference not found: UCUM [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\util\package-info.java:39: warning - Tag @link: reference not found: CommonUnits [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:58: warning - Tag @link: reference not found: org.org.unitsofmeasurement.impl.system.SI SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:58: warning - Tag @link: reference not found: org.org.unitsofmeasurement.impl.system.SI SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:58: warning - Tag @link: reference not found: org.org.unitsofmeasurement.impl.system.SI SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:58: warning - Tag @link: reference not found: org.org.unitsofmeasurement.impl.system.SI SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\AbstractUnit.java:58: warning - Tag @link: reference not found: org.org.unitsofmeasurement.impl.system.SI SI [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\MeasurementFormat.java:200: warning - Tag @link: reference not found: CompoundUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\format\QuantityFormat.java:200: warning - Tag @link: reference not found: CompoundUnit [WARNING] \User\git\unit-ri\src\main\java\org\unitsofmeasurement\ri\model\QuantityDimension.java:110: warning - Tag @link: reference not found: DimensionService |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 28/Sep/16 ] |
| RI does not target Java SE 8, so out of scope for 1.0 |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-14] [Rename FormatStyle to FormatBehavior](https://java.net/jira/browse/UNITSOFMEASUREMENT-14) Created: 14/May/14  Updated: 14/May/14  Resolved: 14/May/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | design, naming |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The FormatStyle actually defines behavior (locale-sensitive vs. insensitive) rather than a style. Calling it FormatBehavior seems more appropriate. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/May/14 ] |
| Renamed |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-13] [Do we need AbstractMeasurement in RI?](https://java.net/jira/browse/UNITSOFMEASUREMENT-13) Created: 14/May/14  Updated: 29/Sep/14  Resolved: 29/Sep/14 | |
| **Status:** | Closed |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | Unassigned |
| **Resolution:** | Complete | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-12~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-12) | RI Generic type issues with Java 8 | Resolved | | is related to | [~~UNITSOFMEASUREMENT-56~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-56) | Could we move AbstractQuantity into A... | Resolved | |  |
| **Tags:** | design, implementation, question |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The Measurement interface is the general abstraction of a Quantity. Allowing even non-numeric or complex type values (e.g. Image or Video) in certain use cases. However, Quantity specializes, thus extends Measurement, so abstract base types like AbstractQuantity and everything extending it also implements Measurement.  Do we want to keep a base class like AbstractMeasurement in the RI or leave it to 3rd party libraries, either at unitsofmeasurement.org or elsewhere? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 29/Sep/14 ] |
| Let's keep it, and if some "privacy" is required, we can move it around and make it not as public as the interface. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 29/Sep/14 ] |
| "Yes" is the answer to the question |

|  |  |
| --- | --- |
| **[Java 8 issues](https://java.net/jira/browse/UNITSOFMEASUREMENT-10)** ([UNITSOFMEASUREMENT-10](https://java.net/jira/browse/UNITSOFMEASUREMENT-10)) [UNITSOFMEASUREMENT-12] [RI Generic type issues with Java 8](https://java.net/jira/browse/UNITSOFMEASUREMENT-12) Created: 14/May/14  Updated: 08/Aug/14  Resolved: 08/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |
| **Environment:** | Java 8 | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-13~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-13) | Do we need AbstractMeasurement in RI? | Closed | |  |
| **Tags:** | compiler, generics, java8 |
| **Sprint:** | Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Some RI classes, at least AbstractMeasurement, AbstractQuantity and AbstractSystemOfMeasurement use generic type inference in ways, that cause a compilation error under Java SE 8. There's a striking similarity of at least one with this (unsolved, deferred to Java 9!) JDK Bug: <https://bugs.openjdk.java.net/browse/JDK-8028442>  Thus unless the classes would be removed anyway (see AbstractMeasurement) or significant changes to RI classes allow to fix this in a Java 8 and forward way, it should be seen as minor. a) OpenJDK considers the behavior its own BUG and should be able to fix it, b) Java ME 8/CLDC is largely based on Java 7, thus new compiler features or Lambdas, etc. are not supported and an RI or all artifacts built with Java 7 are likely more compatible than using Java SE 8 here.  Details: [INFO] ------------------------------------------------------------- [ERROR] COMPILATION ERROR :  [INFO] ------------------------------------------------------------- [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[421,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#1 of ? upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[427,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[433,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#4 of ? upper bounds: javax.measure.Quantity<capture#5 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[449,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#6 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[525,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#7 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[531,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#8 of ? upper bounds: javax.measure.Quantity<capture#9 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[547,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#10 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[613,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#11 of ? upper bounds: javax.measure.Quantity<capture#12 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[618,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#13 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[623,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#14 of ? upper bounds: javax.measure.Quantity<capture#15 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[628,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#16 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[698,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#17 of ? upper bounds: javax.measure.Quantity<capture#18 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[704,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#19 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[709,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#20 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[714,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#21 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/US.java:[181,54] method addUnit in class org.unitsofmeasurement.ri.util.US cannot be applied to given types; required: U found: org.unitsofmeasurement.ri.AbstractUnit<javax.measure.quantity.Angle> reason: inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Angle upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/US.java:[339,55] method addUnit in class org.unitsofmeasurement.ri.util.US cannot be applied to given types; required: U found: javax.measure.Unit<javax.measure.quantity.Volume> reason: inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Volume upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[434,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#22 of ? upper bounds: javax.measure.Quantity<capture#23 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[527,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#24 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[533,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#25 of ? upper bounds: javax.measure.Quantity<capture#26 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[549,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#27 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[613,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#28 of ? upper bounds: javax.measure.Quantity<capture#29 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[618,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#30 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[623,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#31 of ? upper bounds: javax.measure.Quantity<capture#32 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[628,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#33 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[698,34] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#34 of ? upper bounds: javax.measure.Quantity<capture#35 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[704,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#36 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[709,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#37 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[714,34] incompatible types: inference variable Q has incompatible bounds equality constraints: T upper bounds: javax.measure.Quantity<capture#38 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseQuantity.java:[228,26] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#39 of ? upper bounds: javax.measure.Quantity<capture#40 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseQuantity.java:[241,26] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#41 of ? upper bounds: javax.measure.Quantity<capture#42 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseQuantity.java:[249,34] incompatible types: inference variable Q has incompatible bounds equality constraints: Q upper bounds: javax.measure.Quantity<capture#43 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseQuantity.java:[252,26] incompatible types: inference variable Q has incompatible bounds equality constraints: Q upper bounds: javax.measure.Quantity<capture#43 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseMeasurement.java:[227,26] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#44 of ? upper bounds: javax.measure.Quantity<capture#45 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseMeasurement.java:[240,26] incompatible types: inference variable Q has incompatible bounds equality constraints: capture#46 of ? upper bounds: javax.measure.Quantity<capture#47 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseMeasurement.java:[247,34] incompatible types: inference variable Q has incompatible bounds equality constraints: Q upper bounds: javax.measure.Quantity<capture#48 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseMeasurement.java:[250,26] incompatible types: inference variable Q has incompatible bounds equality constraints: Q upper bounds: javax.measure.Quantity<capture#48 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[242,15] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.MagneticPermeability>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.MagneticPermeability> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.MagneticPermeability upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[256,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Length>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Length> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Length upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[258,50] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Force>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Force> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Force upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[266,50] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Acceleration>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Acceleration> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Acceleration upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[268,59] no suitable method found for multiply(javax.measure.Unit<javax.measure.quantity.Length>) method org.unitsofmeasurement.ri.AbstractUnit.multiply(double) is not applicable (argument mismatch; no instance(s) of type variable(s) Q exist so that javax.measure.Unit<Q> conforms to double) method org.unitsofmeasurement.ri.AbstractUnit.multiply(javax.measure.Unit<?>) is not applicable (argument mismatch; inference variable Q has incompatible bounds equality constraints: javax.measure.quantity.Length upper bounds: javax.measure.Quantity<capture#49 of ?>,javax.measure.Quantity<Q>) method org.unitsofmeasurement.ri.AbstractUnit.multiply(org.unitsofmeasurement.ri.AbstractUnit<?>) is not applicable (argument mismatch; no instance(s) of type variable(s) Q exist so that javax.measure.Unit<Q> conforms to org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[270,44] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Energy>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Energy> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Energy upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[272,56] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.DynamicViscosity>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.DynamicViscosity> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.DynamicViscosity upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[276,59] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.KinematicViscosity>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.KinematicViscosity> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.KinematicViscosity upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[282,63] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.MagneticFieldStrength>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.MagneticFieldStrength> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.MagneticFieldStrength upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[284,60] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.MagnetomotiveForce>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.MagnetomotiveForce> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.MagnetomotiveForce upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[286,49] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Luminance>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Luminance> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Luminance upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[288,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Luminance>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Luminance> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Luminance upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[296,59] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.RadiationDoseAbsorbed>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.RadiationDoseAbsorbed> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.RadiationDoseAbsorbed upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[298,60] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.RadiationDoseEffective>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.RadiationDoseEffective> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.RadiationDoseEffective upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[315,58] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Speed>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Speed> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Speed upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[317,64] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[319,64] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[321,64] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[323,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Volume>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Volume> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Volume upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[325,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Volume>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Volume> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Volume upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[327,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Volume>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Volume> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Volume upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[335,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[364,53] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[366,59] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[368,60] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[370,56] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[399,52] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Speed>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Speed> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Speed upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[401,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[484,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Force>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Force> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Force upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[546,53] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Energy>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Energy> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable Q has incompatible bounds equality constraints: javax.measure.quantity.Energy upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[562,50] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Power>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Power> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Power upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[573,46] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Volume>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Volume> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Volume upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[577,43] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Area> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Area upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[579,63] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Pressure>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Pressure> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Pressure upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[581,57] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.ElectricConductance>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.ElectricConductance> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.ElectricConductance upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[583,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Pressure>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Pressure> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Pressure upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[585,46] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Angle>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.Angle> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.Angle upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[587,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.SolidAngle>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inferred type does not conform to upper bound(s) inferred: javax.measure.Unit<javax.measure.quantity.SolidAngle> upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable (inference variable T has incompatible bounds equality constraints: javax.measure.quantity.SolidAngle upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [INFO] 75 errors  [INFO] ------------------------------------------------------------- [INFO] ------------------------------------------------------------------------ [INFO] BUILD FAILURE [INFO] ------------------------------------------------------------------------ [INFO] Total time: 5.521s [INFO] Finished at: Wed May 14 11:12:36 BST 2014 [INFO] Final Memory: 16M/282M [INFO] ------------------------------------------------------------------------ [ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile (default-compile) on project unit-ri: Compilation failure: Compilation failure: [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[421,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#1 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#2 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[427,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#3 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[433,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#4 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#5 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[449,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#6 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[525,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#7 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[531,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#8 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#9 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[547,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#10 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[613,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#11 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#12 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[618,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#13 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[623,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#14 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#15 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[628,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#16 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[698,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#17 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#18 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[704,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#19 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[709,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#20 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractMeasurement.java:[714,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#21 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/US.java:[181,54] method addUnit in class org.unitsofmeasurement.ri.util.US cannot be applied to given types; [ERROR] required: U [ERROR] found: org.unitsofmeasurement.ri.AbstractUnit<javax.measure.quantity.Angle> [ERROR] reason: inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Angle [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/US.java:[339,55] method addUnit in class org.unitsofmeasurement.ri.util.US cannot be applied to given types; [ERROR] required: U [ERROR] found: javax.measure.Unit<javax.measure.quantity.Volume> [ERROR] reason: inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Volume [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[434,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#22 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#23 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[527,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#24 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[533,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#25 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#26 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[549,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#27 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[613,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#28 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#29 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[618,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#30 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[623,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#31 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#32 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[628,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#33 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[698,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#34 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#35 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[704,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#36 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[709,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#37 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/AbstractQuantity.java:[714,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: T [ERROR] upper bounds: javax.measure.Quantity<capture#38 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseQuantity.java:[228,26] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#39 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#40 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseQuantity.java:[241,26] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#41 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#42 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseQuantity.java:[249,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: Q [ERROR] upper bounds: javax.measure.Quantity<capture#43 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseQuantity.java:[252,26] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: Q [ERROR] upper bounds: javax.measure.Quantity<capture#43 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseMeasurement.java:[227,26] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#44 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#45 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseMeasurement.java:[240,26] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: capture#46 of ? [ERROR] upper bounds: javax.measure.Quantity<capture#47 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseMeasurement.java:[247,34] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: Q [ERROR] upper bounds: javax.measure.Quantity<capture#48 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/BaseMeasurement.java:[250,26] incompatible types: inference variable Q has incompatible bounds [ERROR] equality constraints: Q [ERROR] upper bounds: javax.measure.Quantity<capture#48 of ?>,javax.measure.Quantity<Q> [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[242,15] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.MagneticPermeability>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.MagneticPermeability> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.MagneticPermeability [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[256,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Length>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Length> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Length [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[258,50] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Force>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Force> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Force [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[266,50] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Acceleration>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Acceleration> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Acceleration [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[268,59] no suitable method found for multiply(javax.measure.Unit<javax.measure.quantity.Length>) [ERROR] method org.unitsofmeasurement.ri.AbstractUnit.multiply(double) is not applicable [ERROR] (argument mismatch; no instance(s) of type variable(s) Q exist so that javax.measure.Unit<Q> conforms to double) [ERROR] method org.unitsofmeasurement.ri.AbstractUnit.multiply(javax.measure.Unit<?>) is not applicable [ERROR] (argument mismatch; inference variable Q has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Length [ERROR] upper bounds: javax.measure.Quantity<capture#49 of ?>,javax.measure.Quantity<Q>) [ERROR] method org.unitsofmeasurement.ri.AbstractUnit.multiply(org.unitsofmeasurement.ri.AbstractUnit<?>) is not applicable [ERROR] (argument mismatch; no instance(s) of type variable(s) Q exist so that javax.measure.Unit<Q> conforms to org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[270,44] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Energy>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Energy> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Energy [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[272,56] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.DynamicViscosity>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.DynamicViscosity> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.DynamicViscosity [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[276,59] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.KinematicViscosity>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.KinematicViscosity> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.KinematicViscosity [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[282,63] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.MagneticFieldStrength>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.MagneticFieldStrength> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.MagneticFieldStrength [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[284,60] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.MagnetomotiveForce>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.MagnetomotiveForce> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.MagnetomotiveForce [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[286,49] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Luminance>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Luminance> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Luminance [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[288,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Luminance>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Luminance> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Luminance [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[296,59] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.RadiationDoseAbsorbed>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.RadiationDoseAbsorbed> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.RadiationDoseAbsorbed [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[298,60] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.RadiationDoseEffective>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.RadiationDoseEffective> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.RadiationDoseEffective [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[315,58] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Speed>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Speed> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Speed [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[317,64] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[319,64] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[321,64] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[323,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Volume>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Volume> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Volume [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[325,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Volume>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Volume> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Volume [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[327,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Volume>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Volume> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Volume [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[335,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[364,53] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[366,59] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[368,60] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[370,56] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[399,52] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Speed>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Speed> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Speed [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[401,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[484,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Force>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Force> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Force [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[546,53] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Energy>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Energy> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable Q has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Energy [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<Q>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[562,50] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Power>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Power> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Power [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[573,46] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Volume>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Volume> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Volume [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[577,43] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Area>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Area> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Area [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[579,63] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Pressure>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Pressure> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Pressure [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[581,57] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.ElectricConductance>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.ElectricConductance> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.ElectricConductance [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[583,65] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Pressure>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Pressure> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Pressure [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[585,46] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.Angle>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.Angle> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.Angle [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) [ERROR] /C:/Users/Werner/git/unit-ri/src/main/java/org/unitsofmeasurement/ri/util/UCUM.java:[587,51] no suitable method found for addUnit(javax.measure.Unit<javax.measure.quantity.SolidAngle>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inferred type does not conform to upper bound(s) [ERROR] inferred: javax.measure.Unit<javax.measure.quantity.SolidAngle> [ERROR] upper bound(s): org.unitsofmeasurement.ri.AbstractUnit<?>) [ERROR] method org.unitsofmeasurement.ri.util.UCUM.<U>addUnit(U) is not applicable [ERROR] (inference variable T has incompatible bounds [ERROR] equality constraints: javax.measure.quantity.SolidAngle [ERROR] upper bounds: javax.measure.Quantity<Q>,javax.measure.Quantity<T>) |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 14/May/14 ] |
| No direct dependency, but if AbstractMeasurement was removed or relocated, the number of pain points around SE 8 would be reduced. |

|  |  |
| --- | --- |
| **[Java 8 issues](https://java.net/jira/browse/UNITSOFMEASUREMENT-10)** ([UNITSOFMEASUREMENT-10](https://java.net/jira/browse/UNITSOFMEASUREMENT-10)) [UNITSOFMEASUREMENT-11] [API JavaDoc](https://java.net/jira/browse/UNITSOFMEASUREMENT-11) Created: 14/May/14  Updated: 03/Apr/16  Resolved: 03/Apr/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | [0.9](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16916) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | java8, javadoc |
| **Sprint:** | Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| There are a few references in the API JavaDoc causing problems to the Java 8 JavaDoc.  At the moment Java 7 JavaDoc seems fine. E.g. all ME 8 Embedded JSRs used it for various reasons, thus we are not forced to generate JavaDoc with Java 8 until those isssues are resolved.  This provides an overview of places that don't work. [ERROR] Failed to execute goal org.apache.maven.plugins:maven-site-plugin:3.2:site (default-site) on project unit-api: Error during page generation: Error rendering Maven report: [ERROR] Exit code: 1 - javadoc: warning - Error fetching URL: <http://docs.oracle.com/javase/7/docs/api> [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\UnitSupplier.java:31: error: reference not found [ERROR] \* Returns the unit of this UnitProvider  {@linkplain #getValue() value}  . [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\MeasurementConverter.java:41: warning: no description for @param [ERROR] \* @param unit [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Measurement.java:18: error: invalid end tag: </br> [ERROR] \* </br> Arithmetic methods are provided. At least a runtime error (for some [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Measurement.java:23: error: unknown tag: type [ERROR] \* form. See <type>Unit</type> for more information on the supported units. [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Measurement.java:23: error: unknown tag: type [ERROR] \* form. See <type>Unit</type> for more information on the supported units. [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Quantity.java:23: error: unknown tag: Mass [ERROR] \* <code> Unit<Mass> pound = ... Quantity<Length> size = ... Sensor<Temperature><br> [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Quantity.java:23: error: unknown tag: Length [ERROR] \* <code> Unit<Mass> pound = ... Quantity<Length> size = ... Sensor<Temperature><br> [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Quantity.java:23: error: unknown tag: Temperature [ERROR] \* <code> Unit<Mass> pound = ... Quantity<Length> size = ... Sensor<Temperature><br> [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Quantity.java:24: error: unknown tag: Velocity [ERROR] \* thermometer = ... Vector3D<Velocity> aircraftSpeed = ... </code> [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Unit.java:61: error: reference not found [ERROR] \* @see UnitFormat [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Unit.java:85: error: malformed HTML [ERROR] \* static boolean isAngularVelocity(Unit<?> unit) { [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Unit.java:85: error: bad use of '>' [ERROR] \* static boolean isAngularVelocity(Unit<?> unit) { [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Unit.java:127: error: unknown tag: Velocity [ERROR] \* Unit<Velocity> C = METRE.times(299792458).divide(SECOND).asType(Velocity.class); [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Unit.java:184: error: unknown tag: Angle [ERROR] \* Unit<Angle> RADIAN = ONE.alternate("rad").asType(Angle.class); [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Unit.java:185: error: unknown tag: Force [ERROR] \* Unit<Force> NEWTON = METRE.times(KILOGRAM).divide(SECOND.pow(2)).alternate("N").asType(Force.class); [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\Unit.java:186: error: unknown tag: Pressure [ERROR] \* Unit<Pressure> PASCAL = NEWTON.divide(METRE.pow(2)).alternate("Pa").asType(Pressure.class); [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\UnitTransformer.java:32: error: unknown tag: Dimensionless [ERROR] \* Unit<Dimensionless> DECIBEL = Unit.ONE.transform( [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\UnitTransformer.java:37: error: @param name not found [ERROR] \* @param operation the converter from the transformed unit to this unit. [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\UnitTransformer.java:40: warning: no @param for converter [ERROR] Unit<Q> transform(UnitConverter converter); [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\BiFactory.java:20: error: reference not found [ERROR] \* @see Function [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\Converter.java:17: error: reference not found [ERROR] \* whose functional method is  {@link #convert()}  . [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\Converter.java:23: warning - Tag @link: can't find convert() in javax.measure.function.Converter [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\MeasurementConverter.java:20: error: reference not found [ERROR] \* >functional interface</a> whose functional method is  {@link #to()}  . [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\MeasurementConverter.java:36: warning - Tag @link: can't find to() in javax.measure.function.MeasurementConverter [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\UnitConverter.java:15: error: reference not found [ERROR] \* <p>Instances of this class are obtained through the  {@link Unit#getConverterTo(Unit)}  method.</p> [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\UnitSupplier.java:35: warning - Tag @linkplain: can't find getValue() in javax.measure.function.UnitSupplier [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\UnitTransformer.java:17: error: reference not found [ERROR] \* whose functional method is  {@link #transform()}  . [ERROR] ^ [ERROR] \User\git\unit-api\core\src\main\java\javax\measure\function\UnitTransformer.java:27: warning - Tag @link: can't find transform() in javax.measure.function.UnitTransformer [ERROR] \User\git\unit-api\format\src\main\java\javax\measure\format\Parser.java:32: warning: no @param for input [ERROR] O parse(I input) throws ParserException; [ERROR] ^ [ERROR] \User\git\unit-api\format\src\main\java\javax\measure\format\Parser.java:32: warning: no @return [ERROR] O parse(I input) throws ParserException; [ERROR] ^ [ERROR] \User\git\unit-api\format\src\main\java\javax\measure\format\Parser.java:17: error: reference not found [ERROR] \* >functional interface</a> whose functional method is  {@link #parse()}  . [ERROR] ^ [ERROR] \User\git\unit-api\format\src\main\java\javax\measure\format\Parser.java:28: warning - Tag @link: can't find parse() in javax.measure.format.Parser [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\Acceleration.java:21: error: reference not found [ERROR] \* @see Velocity [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\AmountOfSubstance.java:20: error: invalid uri: " <http://en.wikipedia.org/wiki/Amount_of_substance>" [ERROR] \* @see <a href=" <http://en.wikipedia.org/wiki/Amount_of_substance>">Wikipedia: Amount of Substance</a> [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\Angle.java:22: error: reference not found [ERROR] \* @see AngularVelocity [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\AngularAcceleration.java:21: error: reference not found [ERROR] \* @see AngularVelocity [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\Length.java:26: error: reference not found [ERROR] \* @see Velocity [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\Time.java:21: error: reference not found [ERROR] \* @see Velocity [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\Time.java:22: error: reference not found [ERROR] \* @see AngularVelocity [ERROR] ^ [ERROR] \User\git\unit-api\util\src\main\java\javax\measure\util\Range.java:12: error: self-closing element not allowed [ERROR] \* Range limits MUST be presented in the same scale and have the same unit as measured data values.<br/>The Range MUST be immutable. [ERROR] ^ [ERROR] \User\git\unit-api\util\src\main\java\javax\measure\util\TimedData.java:15: error: self-closing element not allowed [ERROR] \* TimedData is a container for a data value that keeps track of its age. This class keeps track of the birth time of a bit of data, i.e. time the object is instantiated.<br/>The TimedData MUST be immutable. [ERROR] ^ [ERROR] \User\git\unit-api\util\src\main\java\javax\measure\util\TimedData.java:34: error: @param name not found [ERROR] \* @param data The value of the TimedData. [ERROR] ^ [ERROR] \User\git\unit-api\util\src\main\java\javax\measure\util\TimedData.java:37: warning: no @param for value [ERROR] protected TimedData(T value, long time) { [ERROR] ^ [ERROR] \User\git\unit-api\service\src\main\java\javax\measure\service\DimensionService.java:19: error: unexpected content [ERROR] \* @see  {@link javax.measure.Dimension} [ERROR] ^ [ERROR] \User\git\unit-api\service\src\main\java\javax\measure\service\DimensionService.java:22: warning - Tag @see:illegal character: "123" in "{@link javax.measure.Dimension}  " [ERROR] \User\git\unit-api\service\src\main\java\javax\measure\service\DimensionService.java:22: warning - Tag @see:illegal character: "64" in "  {@link javax.measure.Dimension}  " [ERROR] \User\git\unit-api\service\src\main\java\javax\measure\service\SystemOfUnitsService.java:30: error: malformed HTML [ERROR] \* Returns the default <a [ERROR] ^ [ERROR] \User\git\unit-api\service\src\main\java\javax\measure\service\SystemOfUnitsService.java:31: error: bad use of '>' [ERROR] \* href=<http://en.wikipedia.org/wiki/International_System_of_Units>"> [ERROR] ^ [ERROR] \User\git\unit-api\service\src\main\java\javax\measure\service\SystemOfUnitsService.java:32: error: unexpected end tag: </a> [ERROR] \* International System of Units</a>. [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\package-info.java:34: error: unknown tag: Temperature [ERROR] \* Sensor<Temperature> sensor ...;<br> [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\package-info.java:38: error: unknown tag: Velocity [ERROR] \* Unit<Velocity> = metrePerSecond = METRE.divide(SECOND);<br> [ERROR] ^ [ERROR] \User\git\unit-api\quantity\src\main\java\javax\measure\quantity\package-info.java:39: error: unknown tag: Velocity [ERROR] \* Vector3D<Velocity> aircraftSpeed = new Vector3D(200.0, 50.0, -0.5, metrePerSecond); |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-10] [Java 8 issues](https://java.net/jira/browse/UNITSOFMEASUREMENT-10) Created: 14/May/14  Updated: 28/Sep/16  Resolved: 28/Sep/16 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-57~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-57) | Latest Java 7 and 8 compilers break M... | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-11](https://java.net/jira/browse/UNITSOFMEASUREMENT-11) | [API JavaDoc](https://java.net/jira/browse/UNITSOFMEASUREMENT-11) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-12](https://java.net/jira/browse/UNITSOFMEASUREMENT-12) | [RI Generic type issues with Java 8](https://java.net/jira/browse/UNITSOFMEASUREMENT-12) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-15](https://java.net/jira/browse/UNITSOFMEASUREMENT-15) | [RI JavaDoc](https://java.net/jira/browse/UNITSOFMEASUREMENT-15) | Sub-task | Resolved | keilw | |  |
| **Tags:** | java8 |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Final Release |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This is an umbrella for Java 8 related bugs or problems |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-9] [Mass Density Unit](https://java.net/jira/browse/UNITSOFMEASUREMENT-9) Created: 13/May/14  Updated: 08/Jun/15  Resolved: 08/Jun/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199), [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8-RC3](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17310) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Improvement | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | blocks | [~~UNITSOFMEASUREMENT-100~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-100) | Define our criterion for inclusion / ... | Resolved | |  |
| **Tags:** | quantity, question, vote |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | Q2/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The UOMo Forum (based on Unit-API 0.6 which this question refers to) was asked the following question:  Von: "Derek Vasconcelos" <forums-noreply@eclipse.org> Betreff: Mass Density Unit Datum: Mittwoch, 05. Februar 2014 15:11  I noticed that the mass density unit(kg/m³, rho symbol) is not present in SI class, is there a specific reason for that?  Thanks |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/May/14 ] |
| My reply from Eclipse forum:  Thanks for the suggestion. I am pretty certain, this unit should be part of  the UCUM (Unified Code for Units of Measure) catalogue, but for reasons  mentioned here at Wikipedia: <http://en.wikipedia.org/wiki/SI#Derived_units> the quantity package does not contain a unit typ for "Mass Density".  While the (quantity) spec module is likely to be restricted to those in the list above extensions and libraries defining additional  quantities may contain such a quantity type if required.  Actually, as he referred to the SI unit system class, without defining an explicit quantity type, such unit could be in a custom unit system, but probably not in the SI/Metric RI class. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 05/Sep/14 ] |
| If this is part of SI units we want to support, we should consider it for API/RI, otherwise it could be an extension via UOM Libraries |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/May/15 ] |
| It's in Table 2 of <http://physics.nist.gov/cuu/Units/units.html>, so wherever we include these units and quantities, Mass Density should also be part of it. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/May/15 ] |
| Actually we already have it under VolumetricDensity, so either we could rename it or keep it (and close this issue) |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/May/15 ] |
| <http://en.wikipedia.org/wiki/Density> calls it "Volumetric Mass Density" to be most precise. <http://physics.nist.gov/cuu/Units/units.html> simply calls it "Mass Density" WDYT? |
| Comment by [dautelle](https://java.net/jira/secure/ViewProfile.jspa?name=dautelle) [ 01/Jun/15 ] |
| +1 MassDensity |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 01/Jun/15 ] |
| I think we should keep in mind that the international reference for units of measurement is BIPM rather than NIST. In my understanding, NIST is specific to one country (USA), while BIPM defines the international standard.  BIMP lists some derived units there: <http://www.bipm.org/en/publications/si-brochure/section2-2.html>. For the density, both "density" and "mass density" are listed by BIPM as acceptable terms. Interestingly, the French translation is "masse volumique", which is emphasis on a different aspect. Maybe the English "Volumetric density" expression is derived from that French translation?  I wonder, isn't the plain "density" word widely understood as "mass density"? Is the risk of confusion really high?  My vote would be:  1. (my preference) Density 2. (second choice) Mass density  I would like to avoid the other choices since they are not listed by BIPM. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 01/Jun/15 ] |
| I'm ok with Density, as I agree with Martin: to me Density implies Mass Density. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 01/Jun/15 ] |
| +1 MassDensity  See <http://en.wikipedia.org/wiki/Density> clearly states "This article is about mass density. For other uses, see <http://en.wikipedia.org/wiki/Density_(disambiguation>)" quoting at least a Dozen other "\*Density" terms in physics alone. Simply calling it Density bears the risk of confusion with either of these. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 02/Jun/15 ] |
| Yes, but note also that the wiki page is named "Density". For what it is worth, we can see also that a Google search of:   * physics "mass density" gives 537,000 hits * physics mass density gives 115,000,000 hits (note the absence of quote). A quick view of the first hits show that they are really talking about the quantity of kg/m3.   So I think that "density" is widely understood as "mass density", even in physics books at the universities. I think that this is similar to PlaneAngle versus SolidAngle: we can remove the Plane part because Angle is widely understood as "plane angle". Keeping PlaneAngle was even maybe a risk of confusing more than helping the users, since many of them would probably search for a plain "angle". |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 02/Jun/15 ] |
| Well we deviated from the most distinct table of 22 named units and quantities (<http://www.bipm.org/en/publications/si-brochure/section2-2.html#section2-2-2> sorry still need to click "Table 3" their links suck ;-,/ ) by doing so. On the other hand as a source of reason (and quite important for e.g. JDK, Eclipse, etc. in many other areas) also calling it "Angle" would be Unicode CLDR and ICU4J. CLDR does not care about either "\*density" unit type so far.  A "salomonic" verdict could be to simply drop it and reduce the number of quantities. Please also take that into consideration. Depends on the real life value of having such quantity in the API. The 7+22 (+2 more including Dimensionless and AngularSpeed which is referred to by GeoAPI) are clear, while in "Table 2" only Area, Volume, Speed and maybe Acceleration feel very common.  Take: "AmountConcentration" or simply "Concentration" (mol/m3) (a) In the field of clinical chemistry this quantity is also called substance concentration. That sounds more like subject to uom-domain or uom-health than the core API |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Jun/15 ] |
| Renamed based on vote. |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-8] [Could remove BiFactory from API](https://java.net/jira/browse/UNITSOFMEASUREMENT-8) Created: 09/May/14  Updated: 18/Sep/14  Resolved: 18/Sep/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-7~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-7) | Getting rid of java.lang.reflect | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-32~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-32) | Remove functional interfaces that are... | Resolved | |  |
| **Tags:** | design, portability, question |
| **Epic Link:** | [Design](https://java.net/jira/browse/UNITSOFMEASUREMENT-27) |
| **Sprint:** | JavaOne |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| If QuantityFactory is moved from RI, this is currently the only class implementing BiFactory. Moving the interface to the module that contains QuantityFactory (or its service/interface part) would decrease the size of the core API. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Aug/14 ] |
| Part of it covered by functional interface issue. Will be resolved there |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 13/Aug/14 ] |
| The Question of QuantityFactory (by the name of BiFactory or more concrete QuantityFactory) being either on an API or implementation level should be discussed and resolved here. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/Aug/14 ] |
| Assigned Otavio to follow up on his suggestion, QuantityFactory could be defined as interface on an API level. Whether to rename BiFactory into a more specific interface like QuantityFactory (which are currently classes in RI or similar implementations) or stick with concrete classes per implementation, we hope to clarify here. |
| Comment by [desruisseaux](https://java.net/jira/secure/ViewProfile.jspa?name=desruisseaux) [ 24/Aug/14 ] |
| I support the proposal to define QuantityFactory as an interface and remove BiFactory. |
| Comment by [otaviojava](https://java.net/jira/secure/ViewProfile.jspa?name=otaviojava) [ 17/Sep/14 ] |
| Done: <https://github.com/JavaMoney/jsr354-ri/pulls> <https://github.com/unitsofmeasurement/unit-api/pull/12> |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 17/Sep/14 ] |
| Unfortunately there is no replacement as the task was about, only BiFactory deleted, that is not the idea, it was similar to merging UnitConverter with another functional interface, so QuantityFactory or something similar should be in API first |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 18/Sep/14 ] |
| Refocussed on changing name |

|  |  |
| --- | --- |
| **[Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)** ([UNITSOFMEASUREMENT-3](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)) [UNITSOFMEASUREMENT-7] [Getting rid of java.lang.reflect](https://java.net/jira/browse/UNITSOFMEASUREMENT-7) Created: 09/May/14  Updated: 30/Oct/15  Resolved: 30/Oct/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Attachments:** | TesteResourceBundle.7z |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | depends on | [~~UNITSOFMEASUREMENT-91~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-91) | Add isSystemAvailable(), getAvailable... | Resolved | | blocks | [~~UNITSOFMEASUREMENT-8~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-8) | Could remove BiFactory from API | Resolved | | **Related** | | | | | is related to | [~~UNITSOFMEASUREMENT-86~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-86) | Getting rid of java.util.Locale and R... | Resolved | |  |
| **Tags:** | portability, reflection |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Mainly QuantityFactory uses Reflection. Which is not available in ME/CLDC. Try move it into separate module or external library outside RI. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Aug/14 ] |
| The remaining RI class to resolve from reflection is org.unitsofmeasurement.ri.format.SymbolMap in the factory/constructor. If a different ME compliant way of handling ResourceBundles can be found, or SymbolMap replaced with an appropriate replacement in the relevant format implementations, there's no further reflection in the RI. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 05/Sep/14 ] |
| My ME SDK has a localization support option that, when enabled, creates 2 classes and a XML file to support dynamically loading of messages.  So I believe we could go the same route? I attached the project it created. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 05/Sep/14 ] |
| Great, thanks, please do, you should also be among admins to "yank" (as BEA support called it ) the ticket if you want to work on it, or just provide the patch. It's certainly among the more important issues towards EDR to discuss either today or at J1. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 05/Sep/14 ] |
| New project created in Eclipse SDK for ME |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 05/Sep/14 ] |
| Cool, thanks, will have a look at it before the call. CU |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 16/Dec/14 ] |
| Quick comment: you can get away with Class.forName(). But not with Method and Field stuff. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 15/Feb/15 ] |
| Unfortunately we need both, it's not enough to know the class of the unit system here:  Class<?> c = Class.forName(className);  Field field = c.getField(fieldName);  Object value = field.get(null);  if (value instanceof AbstractUnit<?>) {  if (isAlias) {  alias((AbstractUnit<?>) value, symbol);  } else {  label((AbstractUnit<?>) value, symbol);  }  ...  in tec.units.ri.format.SymbolMap.  This is called especially when parsing a string to a Unit, unlike formatting where Locale or ResourceBundle are relatively easy to substitute (most of that is done) here we need to create an instance of types. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 20/Apr/15 ] |
| Werner, did you move past the last comment?  If your resourceBundle already contains the name of a class + a field, why not include everything in the resource bundle, including the class of the field instead of the field name?  Maybe I'm missing the point here, but why do you need the field? If it's static information, just store it inside the resource bundle (another one, or the same one)? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 20/Apr/15 ] |
| I tried a workaround some time ago. I think at least on some occasion, likely the exact same symbol used by 2 or more different units (take "m" for METRE vs. MINUTE or it could be MILE in a different unit system ) it failed, but came close to avoiding this. |
| Comment by [leomrlima](https://java.net/jira/secure/ViewProfile.jspa?name=leomrlima) [ 20/Apr/15 ] |
| Do you have the patch for it? If that's the only "bad" part, maybe we can try something around it? I'd like to see what you had before I try something.  And isn't miles **mi**? |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 10/Jun/15 ] |
| See the code-snipped in <https://github.com/unitsofmeasurement/unit-ri/blob/master/src/main/java/tec/units/ri/format/SymbolMap.java> (line 116ff) instead of relying on a limited number of SystemOfUnit implementations, we should be able to ask SystemOfUnitService or a similar place which systems are known and more importantly the units they hold.  Not sure about prefixes, but at least MetricPrefix in the RI is backed by an enum, thus we could try tapping into methods like values() there, too. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 30/Oct/15 ] |
| Got rid of EBNFUnitFormat in RI |

|  |  |
| --- | --- |
| **[Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)** ([UNITSOFMEASUREMENT-3](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)) [UNITSOFMEASUREMENT-6] [Getting rid of java.text](https://java.net/jira/browse/UNITSOFMEASUREMENT-6) Created: 09/May/14  Updated: 08/Feb/15  Resolved: 08/Feb/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | format, portability |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Classes in java.text like Format are not supported by ME/CLDC. They should be factored out into special sub-modules, leaving the "core" RI without them, or to extending libraries such as the "unit-lib" project. |

|  |  |
| --- | --- |
| **[Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)** ([UNITSOFMEASUREMENT-3](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)) [UNITSOFMEASUREMENT-5] [Getting rid of java.math](https://java.net/jira/browse/UNITSOFMEASUREMENT-5) Created: 09/May/14  Updated: 10/Aug/14  Resolved: 10/Aug/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Tags:** | math, portability |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Classes in java.math like BigDecimal are not supported by ME/CLDC. They should be factored out into special sub-modules, leaving the "core" RI without them, or to extending libraries such as the "unit-lib" project. |

|  |  |
| --- | --- |
| **[Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)** ([UNITSOFMEASUREMENT-3](https://java.net/jira/browse/UNITSOFMEASUREMENT-3)) [UNITSOFMEASUREMENT-4] [Getting rid of Serializable](https://java.net/jira/browse/UNITSOFMEASUREMENT-4) Created: 09/May/14  Updated: 11/Mar/16  Resolved: 09/May/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Sub-task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Relates** | | | | | relates to | [~~UNITSOFMEASUREMENT-170~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-170) | Serializable support | Resolved | |  |
| **Tags:** | portability, serialization |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| To improve ME/SE interoperability, all parts of the RI that do not explicitely require it should not use Serializable. |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 09/May/14 ] |
| Removed all usage of Serializable from RI |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-3] [Portability of RI](https://java.net/jira/browse/UNITSOFMEASUREMENT-3) Created: 09/May/14  Updated: 30/Oct/15  Resolved: 30/Oct/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [RI](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15200) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.8](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16915) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Blocks** | | | | | is blocked by | [~~UNITSOFMEASUREMENT-57~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-57) | Latest Java 7 and 8 compilers break M... | Resolved | |  |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-4](https://java.net/jira/browse/UNITSOFMEASUREMENT-4) | [Getting rid of Serializable](https://java.net/jira/browse/UNITSOFMEASUREMENT-4) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-5](https://java.net/jira/browse/UNITSOFMEASUREMENT-5) | [Getting rid of java.math](https://java.net/jira/browse/UNITSOFMEASUREMENT-5) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-6](https://java.net/jira/browse/UNITSOFMEASUREMENT-6) | [Getting rid of java.text](https://java.net/jira/browse/UNITSOFMEASUREMENT-6) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-7](https://java.net/jira/browse/UNITSOFMEASUREMENT-7) | [Getting rid of java.lang.reflect](https://java.net/jira/browse/UNITSOFMEASUREMENT-7) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-18](https://java.net/jira/browse/UNITSOFMEASUREMENT-18) | [Factoring out SE only parts](https://java.net/jira/browse/UNITSOFMEASUREMENT-18) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-86](https://java.net/jira/browse/UNITSOFMEASUREMENT-86) | [Getting rid of java.util.Locale and R...](https://java.net/jira/browse/UNITSOFMEASUREMENT-86) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-119](https://java.net/jira/browse/UNITSOFMEASUREMENT-119) | [Fix L10nPropertyResources](https://java.net/jira/browse/UNITSOFMEASUREMENT-119) | Sub-task | Resolved | leomrlima | | [UNITSOFMEASUREMENT-138](https://java.net/jira/browse/UNITSOFMEASUREMENT-138) | [Correct java.util.logging.Logger usages](https://java.net/jira/browse/UNITSOFMEASUREMENT-138) | Sub-task | Resolved | leomrlima | |  |
| **Tags:** | portability |
| **Epic Link:** | [Portability](https://java.net/jira/browse/UNITSOFMEASUREMENT-21) |
| **Sprint:** | Q1/15, Public Draft |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Umbrella ticket for issues like Serializable, BigDecimal, etc. in the RI and avoiding them for ME compatibility |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-2] [Create Spec Document](https://java.net/jira/browse/UNITSOFMEASUREMENT-2) Created: 29/Apr/14  Updated: 27/Jan/15  Resolved: 27/Jan/15 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [Documentation](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15277), [Spec](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15202) |
| **Affects Version/s:** | [0.4](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&version=16918) |
| **Fix Version/s:** | [0.7.1](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=17107) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Task | **Priority:** | Major |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Σ Remaining Estimate:** | Not Specified | **Remaining Estimate:** | Not Specified |
| **Σ Time Spent:** | Not Specified | **Time Spent:** | Not Specified |
| **Σ Original Estimate:** | Not Specified | **Original Estimate:** | Not Specified |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sub-Tasks:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Key** | **Summary** | **Type** | **Status** | **Assignee** | | [UNITSOFMEASUREMENT-16](https://java.net/jira/browse/UNITSOFMEASUREMENT-16) | [Document Format](https://java.net/jira/browse/UNITSOFMEASUREMENT-16) | Sub-task | Resolved | keilw | | [UNITSOFMEASUREMENT-25](https://java.net/jira/browse/UNITSOFMEASUREMENT-25) | [Adjust spec document to recent code c...](https://java.net/jira/browse/UNITSOFMEASUREMENT-25) | Sub-task | Resolved | keilw | |  |
| **Tags:** | doc, spec |
| **Epic Link:** | [Documents](https://java.net/jira/browse/UNITSOFMEASUREMENT-22) |
| **Sprint:** | August, Q1/15 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This is an umbrella ticket for tasks related to the Spec Document |

|  |  |
| --- | --- |
| [UNITSOFMEASUREMENT-1] [Consider a separate unit-convert module](https://java.net/jira/browse/UNITSOFMEASUREMENT-1) Created: 16/Apr/14  Updated: 08/Jun/14  Resolved: 08/Jun/14 | |
| **Status:** | Resolved |
| **Project:** | [unitsofmeasurement](https://java.net/jira/secure/BrowseProject.jspa?id=13779) |
| **Component/s:** | [API](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&component=15199) |
| **Affects Version/s:** | None |
| **Fix Version/s:** | [0.7](https://java.net/jira/secure/IssueNavigator.jspa?reset=true&mode=hide&sorter/order=ASC&sorter/field=priorityπd=13779&fixfor=16886) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | New Feature | **Priority:** | Minor |
| **Reporter:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) | **Assignee:** | [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) |
| **Resolution:** | Fixed | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue Links:**   |  |  |  |  | | --- | --- | --- | --- | | **Dependency** | | | | | blocks | [~~UNITSOFMEASUREMENT-19~~](https://java.net/jira/browse/UNITSOFMEASUREMENT-19) | Adjust module structure | Closed | |  |
| **Tags:** | conversion, converter, design, enhancement, question |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| While unit conversion is in most cases not as complex as currency conversion (bound by time/date, direction and other factors) there could be devices in need of value acquirement without conversion.  The initial draft contains "functional" elements including UnitTransformer or UnitConverter (plus a more generic base converter) in the core module.  Currently Unit extends both UnitTransformer (which applies a Converter to get a transformed unit) and contains methods like getConverterTo(), thus it has a direct reference to other elements that may not be optional (since Unit isn't UnconvertibleException is also related to these operations of Unit.  Last but not least there is MeasurementConverter used by Measurement/Quantity and their implementations. It provides a simple to(Unit) method to convert a measurement value into another unit.  JSR 275 once contained a "javax.measure.converter" package: <http://jscience.org/jsr-275/api/javax/measure/converter/package-summary.html>  Most of its content was RI only (and a not so clear distinction between API and RI, much like JSR 310 in OpenJDK, among reasons why 275 was turned down) so I can't say, if a separate "converter" (or to be consistent with JSR-354 we may just call it "convert") package and module is worth the effort. MEEP 8 has at least 15 packages mostly optional, so with 5 sub-packages, 4 of them optional under a mandatory root package 363 looks decent so far. This could change slightly if we think tweaking it helps. So the question is, whether factoring out conversion related elements is worth the effort or not?  In some cases an implementing class could implement Unit and e.g. UnitConverter separately, in others it may be necessary to consider a separate Converter or Transformer element independent of Unit or Quantity/Measurement.  WDYT? |

|  |  |
| --- | --- |
| **Comments** |  |

|  |
| --- |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 29/Apr/14 ] |
| The Quantity Pattern by Martin Fowler: <http://martinfowler.com/eaaDev/quantity.html> quoted in the proposal suggests a method like to(Unit) directly with the quantity type, too. While Monetary conversions cannot be simplified as described in Fowler's page in most cases (as seen in JSR 354, some form of "Conversion Context" with additional information like date/time, transaction type, etc.) physical quantities and similar measurement values are fine with that in most cases.  Let's see, what others think about this. |
| Comment by [keilw](https://java.net/jira/secure/ViewProfile.jspa?name=keilw) [ 08/Jun/14 ] |
| See follow-up ticket on modularity |

Generated at Fri Apr 28 20:52:58 UTC 2017 by keilw using JIRA 6.2.3#6260-sha1:63ef1d6dac3f4f4d7db4c1effd405ba38ccdc558.