JAIN SLEE 1.1 Extensions

Table of Contents

Restcomm exposes proprietary extensions to the 1.1 specification, to allow the development of easier or more powerful application code.

The extensions were discussed among multiple vendors, and should become part of the standard in next revision, but there is no guarantee that portability won't be lost when using those.

The extensions source code is available in the Restcomm SLEE Community Git repository, specifically at api/extensions subdirectory. Its javadocs are bundled in the SLEE binary release, in docs/container/javadoc subdirectory. The setup for the source project in Eclipse IDE is similar to the container core, as seen in [_eclipse_ide].

Java archives (JARs) with compiled classes, javadocs and sources are available in the Sonatype Maven Repository at https://oss.sonatype.org/content/groups/public/org/mobicents/servers/jainslee/api/jain-slee-11-ext/

SbbContext Extension

This extension to JAIN SLEE 1.1 introduces *org.mobicents.slee.SbbContextExt* interface, which extends *javax.slee.SbbContext* with methods to retrieve SLEE factories and facilities, avoiding the usage of JNDI context.

```
package org.mobicents.slee;

import javax.slee.ActivityContextInterface;
import javax.slee.Sbb;
import javax.slee.SbbContext;
import javax.slee.facilities.ActivityContextNamingFacility;
import javax.slee.facilities.AlarmFacility;
import javax.slee.facilities.TimerFacility;
import javax.slee.nullactivity.NullActivityContextInterfaceFactory;
import javax.slee.nullactivity.NullActivityFactory;
import javax.slee.profile.ProfileFacility;
import javax.slee.profile.ProfileTableActivityContextInterfaceFactory;
import javax.slee.resource.ResourceAdaptorTypeID;
import javax.slee.serviceactivity.ServiceActivityContextInterfaceFactory;
import javax.slee.serviceactivity.ServiceActivityFactory;

public interface SbbContextExt extends SbbContext {

    public Object getActivityContextInterfaceFactory(
            ResourceAdaptorTypeID raTypeID) throws NullPointerException,
            IllegalArgumentException;
```

```
    public ActivityContextNamingFacility getActivityContextNamingFacility();

    public AlarmFacility getAlarmFacility();

    public NullActivityContextInterfaceFactory
getNullActivityContextInterfaceFactory();

    public NullActivityFactory getNullActivityFactory();

    public ProfileFacility getProfileFacility();

    public ProfileTableActivityContextInterfaceFactory
getProfileTableActivityContextInterfaceFactory();

    public Object getResourceAdaptorInterface(ResourceAdaptorTypeID raTypeID,
            String raEntityLink) throws NullPointerException,
            IllegalArgumentException;

    public SbbLocalObjectExt getSbbLocalObject()
            throws TransactionRequiredLocalException, IllegalStateException,
            SLEEException;

    public ServiceActivityContextInterfaceFactory
getServiceActivityContextInterfaceFactory();

    public ServiceActivityFactory getServiceActivityFactory();

    public TimerFacility getTimerFacility();
}
```

The `getActivityContextInterfaceFactory(ResourceAdaptorTypeID)` *method*

Retrieves the ActivityContextInterface factory for the Resource Adaptor Type with the specified ID.

The `getActivityContextNamingFacility()` *method*

Retrieves the Activity Context Naming Facility.

The `getAlarmFacility()` *method*

Retrieves the Alarm Facility.

The `getNullActivityContextInterfaceFactory()` *method*

Retrieves the Null Activity Context Interface Factory.

The `getNullActivityFactory()` *method*

Retrieves the Null Activity Factor.

The `getProfileFacility()` *method*

Retrieves the Profile Facility.

The `getProfileTableActivityContextInterfaceFactory()` *method*

Retrieves the Profile Table Activity Context Interface Factory.

The `getResourceAdaptorInterface(ResourceAdaptorTypeID,String)` *method*

Retrieves the interface to interact with a specific Resource Adaptor entity, identified by both the entity link name and the Resource Adaptor Type ID.

The `getSbbLocalObject()` *method*

Exposes the SBB local object with the extension interface to avoid unneeded casts.

The `getServiceActivityContextInterfaceFactory()` *method*

Retrieves the Service Activity Context Interface Factory.

The `getServiceActivityFactory()` *method*

Retrieves the Service Activity Factory.

The `getTimerFacility()` *method*

Retrieves the Timer Facility.

ChildRelation Extension

This extension to JAIN SLEE 1.1 introduces the `org.mobicents.slee.ChildRelationExt` interface, which extends `javax.slee.ChildRelation` with methods to create and retrieve SBB entities by name.

```
package org.mobicents.slee;

public interface ChildRelationExt extends ChildRelation {

    public SbbLocalObjectExt create(String name) throws CreateException,
            TransactionRequiredLocalException, SLEEException;

    public SbbLocalObjectExt get(String name)
            throws TransactionRequiredLocalException, SLEEException;

}
```

The `create(String)` *method*

Creates a new SBB entity of the SBB type associated with the relation, with the specified name. The new SBB entity is automatically added to the relationship collection. The returned object may be cast to the required local interface type using the normal Java typecast mechanism. This method is a mandatory transactional method.

The `get(String)` *method*

Retrieves the SBB entity associated with the child relation with the specified name. This method is a mandatory transactional method.

SbbLocalObject Extension

This extension to JAIN SLEE 1.1 introduces the `org.mobicents.slee.SbbLocalObjectExt` interface, which extends `javax.slee.SbbLocalObject` with methods to retrieve the parent SBB Entity, if any, and to also retrieve information such as the child name, and the parent child relation name.

```
package org.mobicents.slee;

public interface SbbLocalObjectExt extends SbbLocalObject {

    public String getChildRelation() throws TransactionRequiredLocalException,
            SLEEException;

    public String getName() throws NoSuchObjectLocalException,
            TransactionRequiredLocalException, SLEEException;

    public SbbLocalObjectExt getParent() throws NoSuchObjectLocalException,
            TransactionRequiredLocalException, SLEEException;
}
```

The `getChildRelation()` method

Retrieves the name of the child relation used to create this object. This method is a mandatory transactional method.

The `getName()` method

Retrieves the name of the object. This method is a mandatory transactional method.

The `getParent()` method

Retrieves the parent SBB object. This method is a mandatory transactional method.

ProfileContext Extension

This extension to JAIN SLEE 1.1 introduces *org.mobicents.slee.ProfileContextExt* interface, which extends *javax.slee.ProfileContext* with methods to retrieve SLEE alarm facility, avoiding the usage of JNDI context.

```
package org.mobicents.slee;

import javax.slee.facilities.AlarmFacility;
import javax.slee.profile.Profile;
import javax.slee.profile.ProfileContext;

public interface ProfileContextExt extends ProfileContext {

    public AlarmFacility getAlarmFacility();

}
```

The `getAlarmFacility()` method

Retrieves the Alarm Facility.

ActivityContextInterface Extension

This simple extension to JAIN SLEE 1.1 introduces *org.mobicents.slee.ActivityContextInterfaceExt* interface, which extends *javax.slee.ActivityContextInterface* with methods to retrieve the timers and names bound to the ACI.

```
package org.mobicents.slee;

import javax.slee.ActivityContextInterface;
import javax.slee.facilities.TimerID;

public interface ActivityContextInterfaceExt extends ActivityContextInterface {

    public TimerID[] getTimers();

    public String[] getNamesBound();

}
```

The `getTimers()` method

Retrieves the IDs of timers currently set which are related to the ACI.

The `getNamesBound()` method

Retrieves the names currently bound to the ACI.

The `suspend()` method

This feature may be used before attaching to an ActivityContextInterface, to ensure that any event fired concurrently will be received. It suspends routing of events in the activity context immediately, till the active transaction ends.

Library References Extension

JAIN SLEE 1.1 standard introduced the Library component, a wrapper for a set of jars and/or classes to be referenced and used by other components types, such as SBBs.

The usage of the standard Library component is very limited, each Library can only refer other Library components. Due to this limitation a developer may not be able to include classes in a Library that depend, just as example, on Resource Adaptor Type interfaces, unless of course those interfaces are in a Library too.

This extension allows libraries to reference other component types, which the developer should use when the classes in the Library need to use classes from that component, by simply extending the JAIN SLEE 1.1 Library Jar XML descriptor.

Extended Library Jar XML Descriptor DTD

The DTD document changes for the extended library jar XML descriptor:

```
<!--
The library element defines a library.  It contains an optional description
about the library, the name, vendor, and version of the library being defined,
zero or more references to any other components that this library
depends on, and information about zero or more jar files that contain
prepackaged classes and resources to be included with the library.

The classes and resources for a library are the sum total of the classes and
resources contained in:
- the library component jar itself (if any)
- the library jars specified by the jar elements (if any)

All these classes are loaded by the same classloader.

Used in: library-jar
-->
<!ELEMENT library (description?, library-name, library-vendor, library-version,
 event-type-ref*, library-ref*, profile-spec-ref*, resource-adaptor-type-ref*,
 sbb-ref*, jar*)>

<!--
The event-type-ref element identifies an event type that the library classes depend.
It contains the name, vendor,and version of the event type.

Used in: library
-->
<!ELEMENT event-type-ref (event-type-name, event-type-vendor,
          event-type-version)>

<!--
The event-type-name element contains the name of an event type referred by
```

```
the library.

Used in: event-type-ref

Example:
    <event-type-name>
        javax.csapi.cc.jcc.JccCallEvent.CALL_CREATED
    </event-type-name>
-->
<!ELEMENT event-type-name (#PCDATA)>

<!--
The event-type-vendor element contains the vendor of an event type referred by
the library

Used in: event-type-ref

Example:
    <event-type-vendor>javax.csapi.cc.jcc</event-type-vendor>
-->
<!ELEMENT event-type-vendor (#PCDATA)>

<!--
The event-type-version element contains the version of an event type referred by
the library

Used in: event-type-ref

Example:
    <event-type-version>1.1</event-type-version>
-->
<!ELEMENT event-type-version (#PCDATA)>

<!--
The profile-spec-ref element identifies an profile specification that the library
classes depend. It contains an optional description about the reference, and the
name, vendor, and version of the referenced profile specification.

Used in: library
-->
<!ELEMENT profile-spec-ref (description?, profile-spec-name,
          profile-spec-vendor, profile-spec-version)>

<!--
The profile-spec-name element contains the name of a profile specification
component.

Used in: profile-spec-ref

Example:
    <profile-spec-name>AddressProfileSpec</profile-spec-name>
```

```
-->
<!ELEMENT profile-spec-name (#PCDATA)>

<!--
The profile-spec-vendor element contains the vendor of a profile specification
component.

Used in: profile-spec-ref

Example:
    <profile-spec-name>javax.slee</profile-spec-name>
-->
<!ELEMENT profile-spec-vendor (#PCDATA)>

<!--
The profile-spec-version element contains the version of a profile
specification component.

Used in: profile-spec-ref

Example:
    <profile-spec-version>1.0</profile-spec-version>
-->
<!ELEMENT profile-spec-version (#PCDATA)>

<!--
The resource-adaptor-type-ref element identifies an resource adaptor type that the
library classes depend. It contains the name, vendor,and version of the RA type.

Used in: library
-->
<!ELEMENT resource-adaptor-type-ref (resource-adaptor-type-name,
          resource-adaptor-type-vendor, resource-adaptor-type-version)>

<!--
The resource-adaptor-type-name element contains the name of a resource
adaptor type component referred by the library.

Used in: resource-adaptor-type-ref

Example:
    <resource-adaptor-type-name>JCC</resource-adaptor-type-name>
-->
<!ELEMENT resource-adaptor-type-name (#PCDATA)>

<!--
The resource-adaptor-type-vendor element contains the vendor of a resource
adaptor type component referred by the library.

Used in: resource-adaptor-type-ref
```

```
Example:
    <resource-adaptor-type-vendor>
        javax.csapi.cc.jcc
    </resource-adaptor-type-vendor>
-->
<!ELEMENT resource-adaptor-type-vendor (#PCDATA)>

<!--
The resource-adaptor-type-version element contains the version of a resource
adaptor type component referred by the library.

Used in: resource-adaptor-type-ref

Example:
    <resource-adaptor-type-version>1.1</resource-adaptor-type-version>
-->
<!ELEMENT resource-adaptor-type-version (#PCDATA)>

<!--
The sbb-ref element identifies an SBB that the library classes depend.
It contains the name, vendor,and version of the SBB.

Used in: library
-->
<!ELEMENT sbb-ref (sbb-name, sbb-vendor,
          sbb-version)>

<!--
The sbb-name element contains the name of a SBB component referred
 by the library.


Used in: sbb-ref

Example:
    <sbb-name>MySbb</sbb-name>
-->
<!ELEMENT sbb-name (#PCDATA)>

<!--
The sbb-vendor element contains the vendor of a SBB component referred
 by the library.

Used in: sbb-ref

Example:
    <sbb-vendor>My Company, Inc.</sbb-vendor>
-->
<!ELEMENT sbb-vendor (#PCDATA)>

<!--
```

```
The sbb-version element contains the version of a SBB component referred
 by the library.

Used in: sbb-ref

Example:
    <sbb-version>1.0</sbb-version>
-->
<!ELEMENT sbb-version (#PCDATA)>

<!--
The ID mechanism is to allow tools that produce additional deployment
information (ie. information beyond that contained by the standard SLEE
deployment descriptors) to store the non-standard information in a separate
file, and easily refer from those tools-specific files to the information in
the standard deployment descriptor.  The SLEE architecture does not allow the
tools to add the non-standard information into the SLEE-defined deployment
descriptors.
-->
<!ATTLIST library-jar id ID #IMPLIED>
<!ATTLIST description id ID #IMPLIED>
<!ATTLIST library id ID #IMPLIED>
<!ATTLIST library-name id ID #IMPLIED>
<!ATTLIST library-vendor id ID #IMPLIED>
<!ATTLIST library-version id ID #IMPLIED>
<!ATTLIST event-type-ref id ID #IMPLIED>
<!ATTLIST event-type-name id ID #IMPLIED>
<!ATTLIST event-type-vendor id ID #IMPLIED>
<!ATTLIST event-type-version id ID #IMPLIED>
<!ATTLIST library-ref id ID #IMPLIED>
<!ATTLIST profile-spec-ref id ID #IMPLIED>
<!ATTLIST profile-spec-name id ID #IMPLIED>
<!ATTLIST profile-spec-vendor id ID #IMPLIED>
<!ATTLIST profile-spec-version id ID #IMPLIED>
<!ATTLIST resource-adaptor-type-ref id ID #IMPLIED>
<!ATTLIST resource-adaptor-type-name id ID #IMPLIED>
<!ATTLIST resource-adaptor-type-vendor id ID #IMPLIED>
<!ATTLIST resource-adaptor-type-version id ID #IMPLIED>
<!ATTLIST sbb-ref id ID #IMPLIED>
<!ATTLIST sbb-name id ID #IMPLIED>
<!ATTLIST sbb-vendor id ID #IMPLIED>
<!ATTLIST sbb-version id ID #IMPLIED>
<!ATTLIST jar id ID #IMPLIED>
<!ATTLIST jar-name id ID #IMPLIED>
<!ATTLIST security-permissions id ID #IMPLIED>
<!ATTLIST security-permission-spec id ID #IMPLIED>
```

This full DTD is available at https://raw.githubusercontent.com/RestComm/jain-slee/master/api/descriptors/library/src/main/resources/slee-library-jar_1_1-ext.dtd

Extended Library Jar XML Descriptor Example

The following XML descriptor examples the definition of references to JAIN SLEE 1.1 Component types other than Library

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library-jar PUBLIC
        "-//Sun Microsystems, Inc.//DTD JAIN SLEE Ext Library 1.1//EN"
        "https://raw.githubusercontent.com/RestComm/jain-
slee/master/api/descriptors/library/src/main/resources/slee-library-jar_1_1-ext.dtd">

<library-jar>
    <library>
        <library-name>extended-library-example</library-name>
        <library-vendor>com.redhat</library-vendor>
        <library-version>1.0</library-version>

        <event-type-ref>
            <event-type-name>ExampleX</event-type-name>
            <event-type-vendor>com.redhat</event-type-vendor>
            <event-type-version>1.0</event-type-version>
        </event-type-ref>

        <library-ref>
            <library-name>LibraryX</library-name>
            <library-vendor>com.redhat</library-vendor>
            <library-version>1.0</library-version>
        </library-ref>

        <profile-spec-ref>
            <profile-spec-name>ProfileX</profile-spec-name>
            <profile-spec-vendor>com.redhat</profile-spec-vendor>
            <profile-spec-version>1.0</profile-spec-version>
        </profile-spec-ref>

        <resource-adaptor-type-ref>
            <resource-adaptor-type-name>ResourceAdaptorTypeX</resource-adaptor-type-
name>
            <resource-adaptor-type-vendor>com.redhat</resource-adaptor-type-vendor>
            <resource-adaptor-type-version>1.0</resource-adaptor-type-version>
        </resource-adaptor-type-ref>

        <sbb-ref>
            <sbb-name>SbbX</sbb-name>
            <sbb-vendor>com.redhat</sbb-vendor>
            <sbb-version>1.0</sbb-version>
        </sbb-ref>

    </library>
</library-jar>
```

Note how the DOCTYPE element is set to the extended DTD instead of the standard one.