OPC UA Java SDK

Migration Guide

From version 2.x to 3.0.0

Table of Contents

This document describes the changes in the SDK design between version 2.x and 3.0.0. You can use this document to help you to migrate your applications built with SDK 2.x to use the new SDK version 3.0.0.

1. Installation

The SDK comes with a new version of the OPC Foundation Java Stack 1.03.

The SDK and Stack jars names have changed slightly. Stack jar is now named:

```
opc-ua-stack-x.y.z-b.jar
```

SDK jar is named (depending on your edition):

```
prosys-opc-ua-java-sdk-client-x.y.z-b.jar (On Client-Binary/Source editions)
prosys-opc-ua-java-sdk-client-server-x.y.z-b.jar (On Client-Server-Binary/Source editions)
prosys-opc-ua-java-sdk-client-server-evaluation-x.y.z-b.jar (On evaluation)
```

Dependencies for the SDK and Stack are the same, however the Apache HttpClient (httpclient-x.x.x.jar) has an additional dependency to Apache Commons Codec (commons-codec-x.x.jar) that was missing in SDK 2.x. It is now added to the SDK distribution packages.

BouncyCastle and SpongyCastle have updated versions.

SpongyCastle is now in a separate folder named 'lib-android' to clarify its usage (note that the libraries do not have the 'sc' prefix anymore).

2. Deployment

No changes (except the ones mentioned in Installation for jar names and the addition of Apache Commons Codec).

3. Stack related

3.1. ByteString Java mapping changed

Stack 1.03 changes the UA ByteString DataType mapping to a concrete `ByteString` class implemented in the Stack instead of mapping to `byte[]`. Internally the data is stored as `byte[]` and instances can be obtained via `ByteString.valueOf(byte...¬)` (it is a varargs method and you can either pass individual values or an array). `ByteString` can be changed back to `byte[]` via `ByteString.getValue()`. Given and returned values are copies, i.e. internal state is immutable.

3.2. Certificate handling

Stack 1.03 adds support for certificate handing and storage. For this reason the related classes are deprecated in the SDK. Stack splits the validation and storage part. The use of

PkiFileBasedCertificateValidator can be replaced by a combination of PkiDirectoryCertificateStore which is then passed on to DefaultCertificateValidator.

4. SDK changes common to Client and Server sides

The generated code for UA Methods has changed for Methods returning more than one output argument. Now a parameter object is generated to the type interface and returned instead of Object[] (or a more specific class if all outputs shared the same superclass).

5. SDK Client side

5.1. 'AsyncResult' generic type now used in generated asynchronous methods

In the SDK version 2.x, the asynchronous methods always returned AsyncResult<Object[]>, now they return the same generic parameters as the normal versions of the methods.

6. SDK Server side

6.1. Changes regarding Optional instantiation

SDK 3.0.0 no longer requires some Optional nodes in InstanceDeclarations to be always instantiated. This means that by default no Optional InstanceDeclarations are instantiated. Therefore, some instances might miss some Optionals if they are instantiated using the default configuration.

The SDK has a new TypeDefinitionBasedNodeBuilderConfiguration class to improve instantiation of types that have a complex InstanceDeclarationHierarchy. It is designed to be immutable after construction via its builder that can be accessed with:

```
TypeDefinitionBasedNodeBuilderConfiguration.builder()
```

It allows the selection of Optionals for instantiation based on the TypeDefinition and BrowseName combination of an InstanceDeclaration.

The default instantiation configuration has been changed from DefaultNodeBuilderConfiguration to TypeDefinitionBasedNodeBuilderConfiguration.builder().build(), i.e. a TypeDefinitionBasedNodeBuilderConfiguration without any Optionals selected.

6.2. Global Discovery Server information model added to bundled models

The Global Discovery Server information model (GDS) is now part of bundled models.

6.3. Generated 'InformationModel' classes renamed to 'ClientInformationModel' and 'ServerInformationModel'

In order to avoid confusion and enforce at compile-time, the InformationModel classes generated by the Code Generator have been renamed into client and server-specific models ClientInformationModel and ServerInformationModel. This avoids accidentally trying to load e.g. a client model to a server.

6.4. Added prefixes to the 'Ids' and 'InformationModel' classes of bundled information models

The new Code Generator allows an optional prefix to be defined for the generated Ids, ClientInformationModel and ServerInformationModel classes. This avoids the need to use fully qualified package names in cases where you need to use the classes from more than one generated namespace. The following prefixes are used, therefore you need to change these (except for the standard namespace that has no prefix):

Namespace	Prefix
http://opcfoundation.org/UA/	(The standard namespace, no prefix)
http://opcfoundation.org/UA/DI/	Di
http://opcfoundation.org/UA/ADI/	Adi
http://PLCopen.org/OpcUa/IEC61131-3/	Plc
http://opcfoundation.org/UA/GDS/	Gds

Therefore e.g. the Device Integration model can be registered via:

```
server.registerModel(DiServerInformationModel.MODEL);
```

6.5. Location of bundled information model NodeSet2 XML files

The new Code Generator defines an option that copies the NodeSet2 XML file used in the generation to the same package as the generated server-side classes. The SDK uses this also internally. Therefore, the bundled NodeSets are no longer in the package 'com.prosysopc.ua.server' but instead they are in 'com.prosysopc.ua.types.XXX.server' in which the XXX is the model-specific part (opcua, di, adi, plc or gds). For example, the bundled Device Integration model (DI) can now be loaded via:

```
server.getAddressSpace().loadModel(DiServerInformationModel.class.getResource("Opc.Ua.Di.NodeSet2.xml"
).toURI());
```

7. Code Generator related

SDK 3.0.0 includes a new Code Generator. Please see the Code Generator manual for more information.

In general, the new Code Generator is not compatible with SDK 2.x nor is the 2.x Code Generator compatible with SDK 3.x. Therefore, you need to regenerate any previously generated code using the new Code Generator. The generated method signatures for manual implementation are mostly the same, except:

1. `ByteString`s in place of `byte[]` if a parameter was UA ByteString.
2. Generated UA Methods that have multiple outputs will now return a parameter object instead of `Object[]` (or in the case that all of the outputs have the same type, an array of the respective type).