Getting Started with the Java Adapter

On this Page:

Related Pages:

- Preferences of the xUML Services
- Importing Java™ Classes and Properties Resource Files

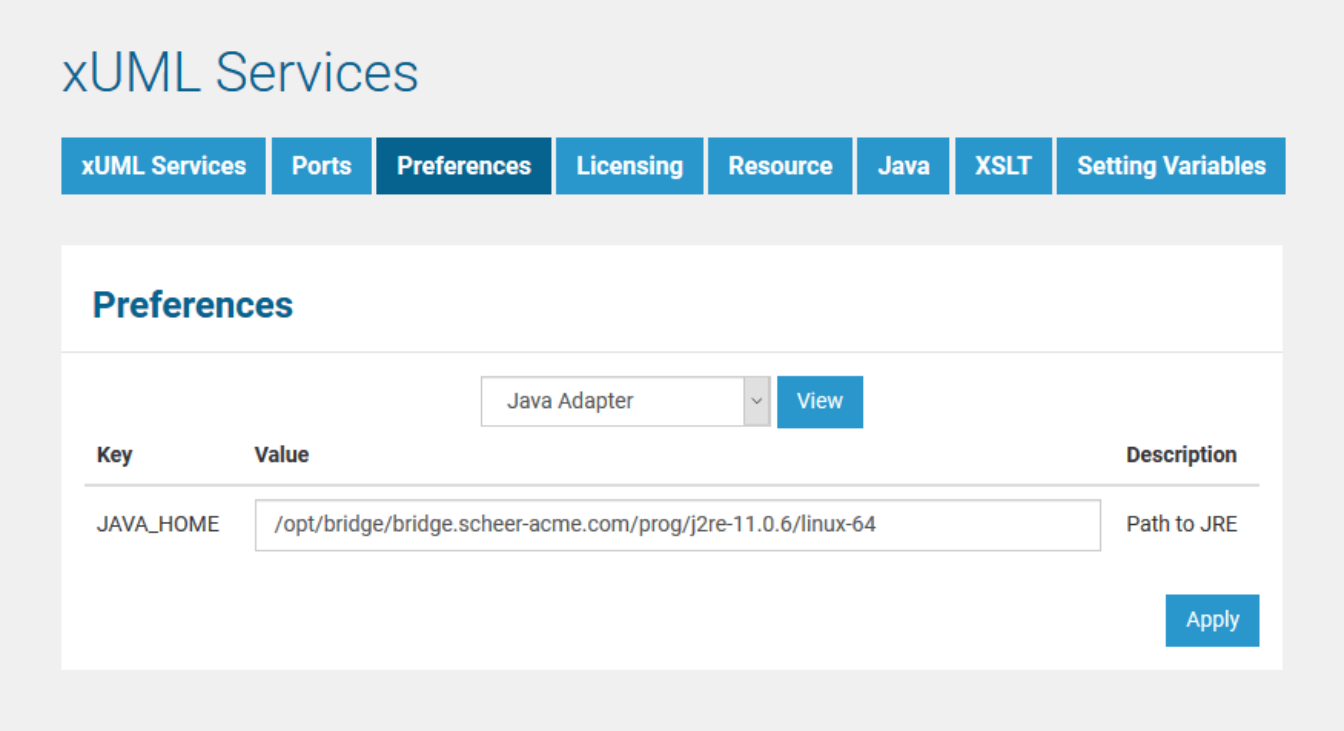
In order to use the Java adapter in your UML model, you need respect some prerequisites.

Java Runtime Environment

To be able to use the Java adapter in your UML model, you need to setup the Java Runtime environment on the Bridge.

On the **Preferences** tab of the Bridge, you can define, which version of the Java runtime you want to use for all deployed composite services. Select preference **Java Adapter** to display the JAVA_HOME path that is actually used.

During Bridge installation, JAVA_HOME is set to the recommended path (JRE delivered with the Bridge). We strongly recommend not to use a Java runtime that is older than the one that has been installed with Bridge. Otherwise, deployed xUML services may not run properly.



Refer to Preferences of the xUML Services for more information on how to set the Java preferences.

Requirements for Importing Java Classes

In order to import the Java classes, you need the JAR files containing all necessary classes and all its referenced classes. All classes need to follow the JavaBeans™ specification.

Especially the following conditions must apply:

Element	Condition
Classes	All classes that will be referenced by the Java adapter need to be declared **public** and have a public default constructor (zero-argument constructor).
Methods	Methods that will be called by the Java adapter need to be declared **public** and must be **static**.
Data Types	For all method parameter and return types, the following conditions need to be fulfilled: • All types can be mapped to a Bridge base type. • They hold a public default constructor (zero-argument constructor). The class attributes have corresponding getter and setter methods. All class attributes are subject to the same conditions that apply to parameter and return types.

The following table shows all possible mappings of Java data types to Bridge base types. The mappings will be prompted during the import of the Java classes, if the xUML Importer cannot identify them automatically.

Java Data Type	Bridge Base Type
boolean, java.lang.Boolean	Boolean
byte, java.lang.Byte	Integer
short, java.lang.Short	Integer
int, java.lang.Integer	Integer
long, java.lang.Long	Integer
float, java.lang.Float	Float
double, java.lang.Double	Float
char, java.lang.Character	String
java.lang.String	String
java.util.Date and subclasses	DateTime
java.util.GregorianCalendar and subclasses	DateTime
java.io.InputStream and subclasses	Blob
byte[], java.lang.Byte[]	Blob
java.util.Collection and implementations	Array
Arrays except byte[] and java.lang.Byte[]	Array
java.util.Map and implementations	Array of a specialized class with stereotype <<MapEntry>> having two attributes of Bridge base type

The **java.io.File** class cannot be mapped, as it is not known, if it will be used as input or output.

If you cannot import a Java class because it does not apply to the requirements listed above, you can write your own Java wrapper class instead without modifying the original class. If the wrapper class applies to the import preconditions, you can import it instead.

Arrays of **InputStream** are not supported.

Java Class Development Hints

Topic	Hint
Checked exceptions in the default constructor	The Bridge Java wrapper cannot handle checked exceptions in the default constructor of a Java class. If you need this nevertheless, you need to wrap the checked exception with an unchecked one.
Data manipulation on the Java object	If you want to manipulate data of a Java object, there is no need to transfer it to xUML and back (if not necessary). Leave the object management to Java and store the created objects in a thread-save concurrent hashmap. Then, simply transfer only an object id between the xUML model and the Java implementation.

Catching exceptions after adapter call	If you want to catch exceptions that occurred during the adapter call in your xUML model, you need to throw these exceptions in your Java code using one of the following:
	
	`throw new BridgeException("your error message", "your error code", "your error domain");`
	`throw new BridgeException("your error message", "your error code");`
	`throw new BridgeException("your error message");`
	
	In order to use `BridgeException`, you need to import class **ch.e2e.bridge.server.BridgeException** from **addon.jar** to your Java implementation. You can find this JAR in your Magic Draw installation folder at /plugins/ch.e2e.builder.plugin.magicdraw/lib/system/addon.jar.
	
	  Example Files (Builder project Add-ons):
	
	<your example path>\Add-ons\JavaServices\uml\javaThrowException.xml