SAP Business Connector Migration Guide

HELP.BCMIDCONF

Release 4.8

Copyright

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help → General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
Example text	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Contents

Document Scope

This document describes the necessary steps for migrating the BC from version 4.6 or 4.7 to version 4.8.

Incompatibilities with Previous Versions

Due to the change of the runtime environment, there are known incompatibilities between BC 4.8 and the previous versions 4.6/4.7 Generally, there are three technical groups of incompatibilities that have an impact on productive usage of the BC:

- **JCo related incompatibilities**
  As Business Connector 4.8 uses JCo 3.0, it inherits at least partially the incompatibilities of JCo 3.0 with respect to JCo 2.1. Some of the incompatibilities are hidden by the Business Connector infrastructure as long as this infrastructure is not bypassed by customer or partner applications running on top of the Business Connector. However, use cases, which bypass the BC infrastructure, are not supported.
- **Java IDoc Library related incompatibility**
  Business Connector 4.8 switches to the Java IDoc Library 3.0, which is incompatible with the Java IDoc Library 1.0. Although the structure of the IDoc interfaces for dealing with an IDoc instance is still the same, the APIs are not source code compatible. But a semi-automatic source code conversion from the 1.0 to the 3.0 API is available in BC 4.8.
- **Platform related incompatibilities**
  Business Connector 4.8 switches to JDK 1.5 and newer operating system versions. JDK 1.5 has incompatibilities to its previous releases. Here some manual adjustments to custom source code may be necessary. Note, that it is also not guaranteed that internal BC APIs don't change. Only BC APIs documented in the public Java Doc are guaranteed to be available again in 4.8. Therefore, you should make sure that your packages use only the documented public APIs of Business Connector, before deploying an existing package on a Business Connector 4.8.
- **Runtime incompatibilites**
  As of BC 4.8 the method `remove(IDataCursor,String)` of class `com.wm.data.IDataUtil` returns `java.lang.Object`; before its return value was `void`. As a result Java classes that utilize the said method in its original form (because they were compiled in BC 4.7 or older) run into an exception of the following kind:
  ```
  java.lang.NoSuchMethodError:
  com.wm.data.IDataUtil.remove(Lcom.wm.data.IDataCursor;Ljava/lang/String;)V
  ```
  If those classes cannot be re-compiled in BC 4.8 for whichever reason (for instance because their source code is not available), then the issue can be fixed only by updating the respective class files (the byte code). This process is explained by note 1340488.

Now let's take a closer look at the different usage types of Business Connector and how a user is facing incompatibilities there:

Installation time related incompatibilities

- SAP Business Connector 4.8 will not run on all platforms on which Business Connector 4.7 has been supported. This is caused by the requirements of the Java runtime and the native libraries used by the BC. (**IC0**)
  Users who like to upgrade from such a platform also need to upgrade/replace their operating system. Moreover, a fresh BC installation followed by a migration of all packages will be necessary in those cases. As a consequence the configuration needs to be transferred as well.

Runtime related incompatibilities

Here we distinguish two cases:

In case 1 we assume that all existing custom BC packages are to be used again after the upgrade and that these packages are already compiled in case they contain Java Services. The user simply wants to continue using the packages without any change.

In case 2 the user invests some time into the migration of his Java Services because of known incompatibilities. Case 2 incompatibilities are the ones for which there is no alternative implementation in 4.8.

Relevant for case 1 only:

- Custom Java Services using JCO.Client directly will no longer work. (**IC1**)
  This is an unsupported use case as those Services bypass the SAP Adapter infrastructure.
- Custom Java Services using the Java IDoc Library 1.0 for mappings will not work. (**IC2**)
  Though the structure of the IDoc interfaces for dealing with an IDoc instance in the Java IDoc Library 3.0 included in Business Connector 4.8 is similar to the Java IDoc Library 1.0, it is nevertheless incompatible because of signature changes – in particular there are new exceptions. At least adding an additional conversion Service step will be necessary, before the mapping can be invoked.
- With JDK 1.5 the sapxmltoolkit is no longer supported. (**IC3**)
  Instead the XSLT engine and the SAX parser included in the JDK need to be used. Normally, this should not be noticed by users except for those who leveraged special features of the sapxmltoolkit not belonging to the standard.
- JDK 1.5 contains incompatibilities that occur only at runtime. (**IC4**)
  For example, BigDecimal.toString() changed its semantics and this could break existing logic that was compiled and tested on previous BC versions. Refer to the following Sun migration guide to find issues in your code:
  http://java.sun.com/j2se/JM_White_Paper_R6A.pdf
- Java Services that use non-public classes of the Business Connector that have been deleted in BC 4.8 will fail with `NoSuchMethodError` or `NoClassDefFoundError`. (**IC5**)
  Unused classes and duplicate implementations have been consolidated in Business Connector 4.8.

Relevant for case 1 and 2:

- Executing function modules that need a GUI attached to the connection – the parameter “Use SAPGUI” is set to On or Hidden – will no longer work. JCo 3.0 does not support starting a SAPGUI. (**IC6**)
  BAPIs and function modules that make use of a GUI and claim to be remotely callable are not well designed and should be fixed anyway. Hence, you should encourage SAP application developers to fix these application function modules.
   Note: starting with Service Release 3, JCo 3.0 (and therefore BC 4.8) will support the „Use SAPGUI“ feature again. For the necessary prerequisites see SAP note 1258724.

- Communication with external RFC servers (written with RFC SDK or JCo standalone) is no longer possible with Business Connector 4.8. (**IC7**)
  JCo 3.0 no longer allows external-external communication, i.e. using the RFC protocol between Non-ABAP systems.

Development and design time related incompatibilities

Users who did their own development for their custom applications on the Business Connector with extensions in their Java Services could experience problems when trying to make fixes and improvements within their coding. Some of the runtime incompatibilities are relevant here as well, but there will be migration possibilities.

- Using classic RFC debugging does not work with JCo 3.0. This is related to **IC6**, as it also needs to start a SAPGUI when the "ABAP Debug" parameter is set to "On". Only external debugging is supported with JCo 3.0 (External debugging is available starting with R/3 Kernel 6.20) (**IC8**)
   Note: starting with Service Release 3, JCo 3.0 (and therefore BC 4.8) will support the „ABAP Debug" feature again. For the necessary prerequisites see SAP note 1258724.
  The runtime of Services communicating with those systems is not affected
- Already with JDK 1.3 the Java compiler has a stricter check regarding the names of classes and names of sub packages of the Java package the class belongs to. Name clashes are no longer allowed. E.g. a class named "foo" in a Java package named "bar" may not be created, if there is a Java package named "bar.foo". As the BC organizes Java Services in a such a way that can lead to these name, Business Connector 4.8 needed to change the Java Service implementation incompatibly in order to allow Services to still be grouped as before (**IC9**).
  As now a JDK 1.5 compiler is used for Service compilation (instead of JDK 1.2), users who structured their Java Services so that name clashes occur, won't be able to recompile those existing Java Services without modification. But as long as they only import the (compiled) Java Services into BC 4.8, without modifying the code, BC 4.8 will be able to execute these Java Services, even if there are JDK 1.5 compile time incompatibilities.
- As already mentioned in **IC2**, the API structure of the Java IDoc Library 3.0 is similar to the one of the Java IDoc Library 1.0, but nevertheless incompatible, because of signature changes – in particular there are new exceptions (**IC10**).
  Custom Java Services providing mappings for IDocs need to be adjusted to the Java IDoc Library 3.0. At least adding an additional conversion Service step will be necessary, before the mapping can be invoked.
- Custom Java Services won't compile, if they face the compile time incompatibilities of Java 5 or are using internal classes of the Business Connector that have been removed with Business Connector 4.8 (see also **IC5**). (**IC11**)
  Custom Java Services need to be adapted to Java 5 and only make use of the public Java API of the Business Connector.

Deprecated Services of Business Connector 4.6/4.7

Since Business Connector 4.6 a set of Services within the SAP Adapter has been deprecated and therefore should have no longer been used. They still existed for runtime compatibility reasons, but have now been removed with Business Connector 4.8. Before importing a package to Business Connector 4.8, you should make sure that all those Services use their replacement.

Previous Service/Specification	New Service/Specification
wm.PartnerMgr.gateway.transport.	pub.sap.transport.ALE:InboundProcess

ALE:InboundProcess	
wm.PartnerMgr.gateway.transport.ALE:OutboundProcess	pub.sap.transport.ALE:OutboundProcess
wm.PartnerMgr.gateway.transport.ALE:getRoutingInfo_Default	pub.sap.transport.ALE:getRoutingInfo_Default
wm.PartnerMgr.gateway.transport.RFC:InboundProcess	pub.sap.transport.RFC:InboundProcess
wm.PartnerMgr.gateway.transport.RFC:OutboundProcess	pub.sap.transport.RFC:OutboundProcess
wm.PartnerMgr.gateway.transport.BAPI:InboundProcess	pub.sap.transport.BAPI:InboundProcess
wm.PartnerMgr.gateway.transport.BAPI:OutboundProcess	pub.sap.transport.BAPI:OutboundProcess
wm.PartnerMgr.gateway.transport.XML:InboundProcess	pub.sap.transport.XML:InboundProcess
wm.PartnerMgr.gateway.transport.XML:OutboundProcess	pub.sap.transport.XML:OutboundProcess
wm.PartnerMgr.gateway.transport.XML:xmlRoutingInfo	pub.sap.transport.XML:xmlRoutingInfo
sap:confirmTID	pub.sap.client:confirmTID
sap:connect	pub.sap.client:connect
sap:createTID	pub.sap.client:createTID
sap.admin:getAttributes	pub.sap.client:getAttributes
sap.admin:getFunctionInterface	pub.sap.client:getFunctionInterface
sap.admin:getStructureDefinition	pub.sap.client:getStructureDefinition
sap:invoke	pub.sap.client:invoke
sap:indirectCall	pub.sap.client:invokeTransaction
sap:lockSession	pub.sap.client:lockSession
sap:releaseSession	pub.sap.client:releaseSession
sap.bapi:encode	pub.sap.bapi:encode
sap.bapi:decode	pub.sap.bapi:decode
sap.bapi.Browser:createTemplate	pub.sap.bapi:createTemplate
sap.bapi.transaction:rollback	pub.sap.bapi:rollback
sap.bapi.transaction:commit	pub.sap.bapi:commit
sap.rfc:encode	pub.sap.rfc:encode

sap.rfc:decode	pub.sap.rfc:decode
sap.rfc:createTemplate	pub.sap.rfc:createTemplate
sap.idoc:encode	pub.sap.idoc:encode
sap.idoc:decode	pub.sap.idoc:decode
sap.idoc:encodeSDATA	pub.sap.idoc:encodeSDATA
sap.idoc:decodeSDATA	pub.sap.idoc:decodeSDATA
sap.idoc:transformFlatToHierarchy	pub.sap.idoc:transformFlatToHierarchy
sap.idoc:transformHierarchyToFlat	pub.sap.idoc:transformHierarchyToFlat
sap.idoc.routing:inbound	pub.sap.idoc.routing:inbound
sap.idoc.routing:inboundDefault	pub.sap.idoc.routing:inboundDefault
sap.idoc.routing:outbound	pub.sap.idoc.routing:outbound
sap.idoc.routing:outboundDefault	pub.sap.idoc.routing:outboundDefault
sap.idoc.routing:registerService	pub.sap.idoc.routing:registerService
sap.idoc.routing:unregisterService	pub.sap.idoc.routing:unregisterService
sap:OutboundRFC	pub.sap.client:invoke *(for synchronous RFC)* *or* pub.sap.client:invokeTransaction *(for tRFC)*
sap:InboundRFC	*No replacment*
sap:disconnect	*No longer necessary, because session pools are used*
sap:tablesToRecordLists	*No longer necessary, because tables can be handled in flow now natively*

Upgrade Procedures Overview

The necessity for upgrade procedures is caused mainly by the incompatibilities introduced by Business Connector 4.8. There is as much tool support as possible, so that most of the migration can be done automatically.

Installation related migration

A user who plans an upgrade often needs a new installation, as the supported platforms for versions 4.6/4.7 on one side and 4.8 on the other side are different. (See **IC0**.) Therefore this migration guide describes in detail what is necessary to transfer custom settings from an existing 4.x installation to a new Business Connector 4.8 installation. A function that archives all important files from a previous installation is also available for customers. For those who can do a true upgrade, the procedure will be as usual: Deinstallation of the old version and afterwards Business Connector 4.8 has to be installed into the same directory.

Custom development related migration

Users who developed Services exclusively with the flow language should not require a migration. Even for Java Services a compatibility mode will ensure that they will work correctly as long as there are no runtime incompatibilities (**IC1** – **IC4**). The most effort needs to be spent on custom Java Services, for which new development is desired or needed:

- Users who developed their own Java Services may have to do a migration of their Java Services, if they want to compile them with JDK 1.5. (see **IC9**) This can be done with the jcode utility, which is included in all Business Connector installations. With that tool it is possible to create code fragments of Java Services on a Business Connector 4.7 and put them together again with a Business Connector 4.8 jcode utility to a Java source file that represents a Business Connector 4.8 Java Service without name clashes, which can be compiled afterwards. A detailed description and the constraints to consider are included in this guide.
- Users who developed IDoc mappings using the Java IDoc Library included in Business Connector 4.7 need to adjust their Services. Business Connector 4.8 provides two possibilities to migrate these Services:
    - Business Connector 4.8 includes the Java Base IDoc Library 1.0 and provides a wrapper for the Java IDoc Library 3.0 that implements the Java Base IDoc Library 1.0 interfaces. Services that can convert between the formats are provided so that existing mappings still work after a pretty simple adjustment of the containing Flow Service.
    - An upgrade/migration tool that supports customers to migrate their Java mappings based on the Java IDoc Library 1.0 to the Java IDoc Library 3.0 is provided. This tool does the conversions that can be done automatically. The rest – the more sophisticated coding – has to be converted manually.
- Last but not least users who developed Java Services will have to check their coding manually for JDK 1.5 compile time and runtime incompatibilities (see **IC4** and **IC11**) (e.g. **`enum`** being a new keyword) as soon as they migrate their Business Connector package to 4.8. This guide will describe the known issues and can thus help users identifying such incompatibilities in their coding.

Upgrade Procedures Detail

Target OS is equal to Source OS

If you have installed your BC 4.6 / 4.7 on an OS which is supported by BC 4.8, you can proceed as follows:

- Shutdown the old BC.
- Backup your whole existing BC directory.
- Deinstall your BC (this will keep files which have been created/modified after installation, meaning it will keep your config changes and your packages!).
- Do NOT delete this BC directory; do not delete left-over files in it!
- Install the new BC 4.8 version into this directory.
- Check if you have made some changes to bin/server.bat or server.sh in your old BC. If yes, reapply these changes (except JDK-settings!) for 4.8.
- Start the new BC.

When starting, BC will now automatically load all existing config files (they are compatible between 4.6 – 4.8) and packages.

Proceed with Check if packages have been imported correctly

Target OS is not equal to Source OS

Most users will face this situation. They installed BC 4.7 e.g. on a Windows 2000 machine, which is not supported by BC 4.8 anymore. This means you will first do a new installation on the new OS (e.g. Windows Server 2003). Then you will perform the migration steps as described below.

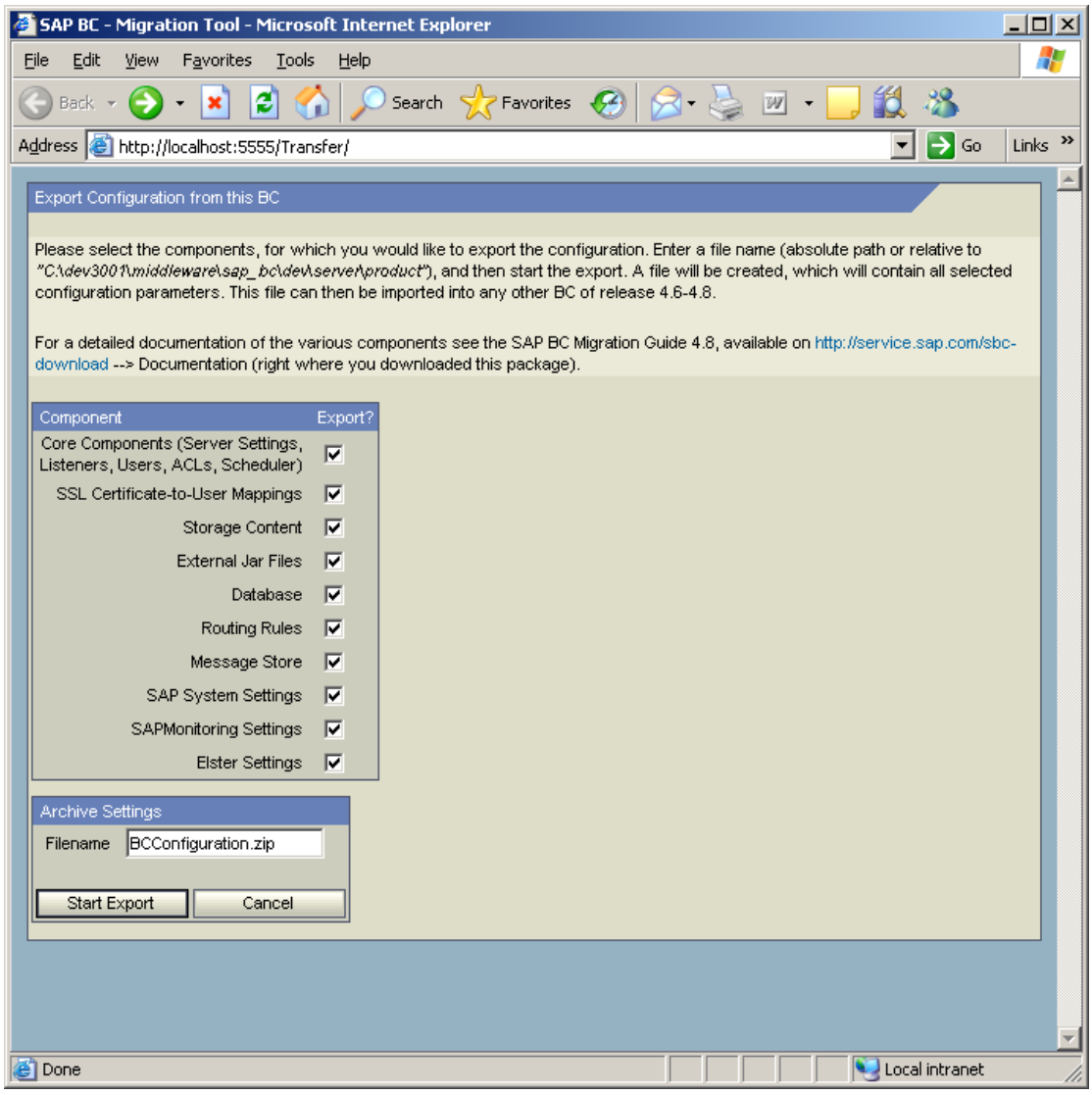
	Before you start, please make sure both BCs, the old and the new one, are currently idle and not processing any transaction.

Import configuration from old BC to new BC

- Login to the old BC and from “Packages → Management” export all your custom packages by pressing the “Archive” button behind the package. **Do not** export any of the Wm* or SAP* packages, because these are internal BC packages which have changed in BC 4.8! The corresponding zip files will be created in sapbc4x/Server/replicate/outbound. (Note: if you still have the original zip files that were installed on this BC, and no changes have been done to the corresponding packages meanwhile, you can of course omit this step and use those files.
- Install the new BC on the new OS. Then copy the files from the previous step into sapbc48/Server/replicate/inbound, start the new BC and import all these packages from “Packages → Management → Install Inbound Releases”.
  If you had installed the packages ELSTER_EXT and/or SAPMonitoring on your old BC, install these packages as well. (Note that a new version of SAPMonitoring is

necessary for SAP BC 4.8. Download it from the Service Marketplace: "http://service.sap.com/sbc-download → CCMS Monitoring Package" and follow the installation instructions.)

- Now shut down the new BC. Don't do any customzing on it yet!
- Download the "Transfer" package (available as separate download on the Business Connector download page in the Service Marketplace: http://service.sap.com/sbc-download → Documentation (right where you downloaded this Migration Guide).
- Import the "Transfer" package into your old BC installation (copy the package file to your replicate/inbound directory and then goto "Packages → Management → Install Inbound Releases" in the Administrator Server UI). There are prerequisites regarding the installed CoreFix, when migrating configurations from 4.6 or 4.7 using the "Transfer" package:
  - Business Connector 4.6: CoreFix 6
  - Business Connector 4.7: CoreFix 4
- Then go back to "Packages → Management" and click on the "Home" icon of the Transfer package. The following screen will open:



- Note that the screen could look slightly different in your case, depending on which components of the Business Connector you are using. For example, if you never defined any database alias in your BC, then the component "Database" will not be

active. And similarly, if your “Message Store” is empty or you don’t have the SAPMonitoring package installed, then these components will not be active.

- De-select those components, whose customizing you don’t want to import into your new BC. Usually you will want to import everything except for possibly “Storage Content” and “Message Store”.
  “Storage Content” means application data that custom packages may persist using the “pub.storage” Services from WmPublic. Usually this is only temporary data, but if your application uses this store for permanent data, select this component.
  The “Message Store” contains status information and payload data (depending on your WmPartners settings) for Transactions, e.g. IDocs. Usually this is also not needed on the new BC, but if you want to continue monitoring the IDoc status of already processed IDocs, select this component. Make sure you have no transaction traffic to/from SAP while performing the following steps. E.g. you can stop the SAP listener(s). Note that because this data can be quite a lot, it is put into a second zip file with the name “MsgStore_ + <Filename>”.
- Change the Filename, if you desire a different one. You can also enter an absolute path here, if you don’t want the zip file(s) to be created in the BC installation directory (sapbc4x/Server).
- Press “Start Export”

If you selected “Message Store” and there is a large number of transactions currently stored in the BC, the export may take a while. Once it is finished, you can find the customizing data in the given zip file (and the Message Store content in a file with the prefix “MsgStore_” in the same directory).

- Now copy the zip file (or the two zip files) over to the new BC machine and unzip them there into the new BC’s installation directory (e.g. sapbc48/Server). Make sure that unzip uses the folder information in the zip file!
- Start the new BC

The new BC has now imported the configuration. One point, however, deserves special attention: if in the old BC you have used Outbound SSL certificates or Elster certificates and these certificate files were located outside of the BC installation directory, then they have been added to the zip file with absolute paths. For example let’s suppose the old BC was installed in C:\sapbc47\Server and the certificate files were located in a directory D:\certificateStore, then the exported configuration zip file will contain the absolute path information “D:\certificateStore”, and if there is no D: drive on the new machine, the unzip process will fail. In this case you need to perform a bit of manual work:

- Open the zip file e.g in WinZip.
- Select these certificate files and extract them to a directory of your choice **without using path information!**
- Delete these certificate files from the zip file, and then unzip it into the sapbc48/Server directory as usual (using path information).
- For the Outbound SSL certificates open the file sapbc48/Server/config/server.cnf in a text editor and adjust the three parameters
  `watt.security.signedCert`
  `watt.security.caCert`
  `watt.security.privateKey`
  to reflect the new location of these files.
- For the Elster SSL certificates open the file sapbc48/Server/packages/ELSTER_EXT/config/settings.cnf in a text editor and change the lines
  `<value name="pkcs12File">…</value>`

and
`<value name="recipientCertificateFile">…</value>`
to reflect the new location of these files.

- Now finally start the new BC.

Check if packages have been imported correctly

In the previous steps you have imported the configuration and the packages into the new BC or you have installed the new BC in the same directory so that the old configuration and packages will be used automatically.

Now please check if there are any problems with loading the packages:

- In the new BC, go to “Packages → Management”.
- Check if any packages could not be loaded successfully.

If some of the Services could not be loaded because of NoClassDefFoundErrors, you may have to add dependencies to your package in BC Developer. In 4.8, some DB classes were moved from the server to WmDB. So if you are using for example the class JDBCConnection in a Java Service, you will have to add a dependency to WmDB package version 2.0.

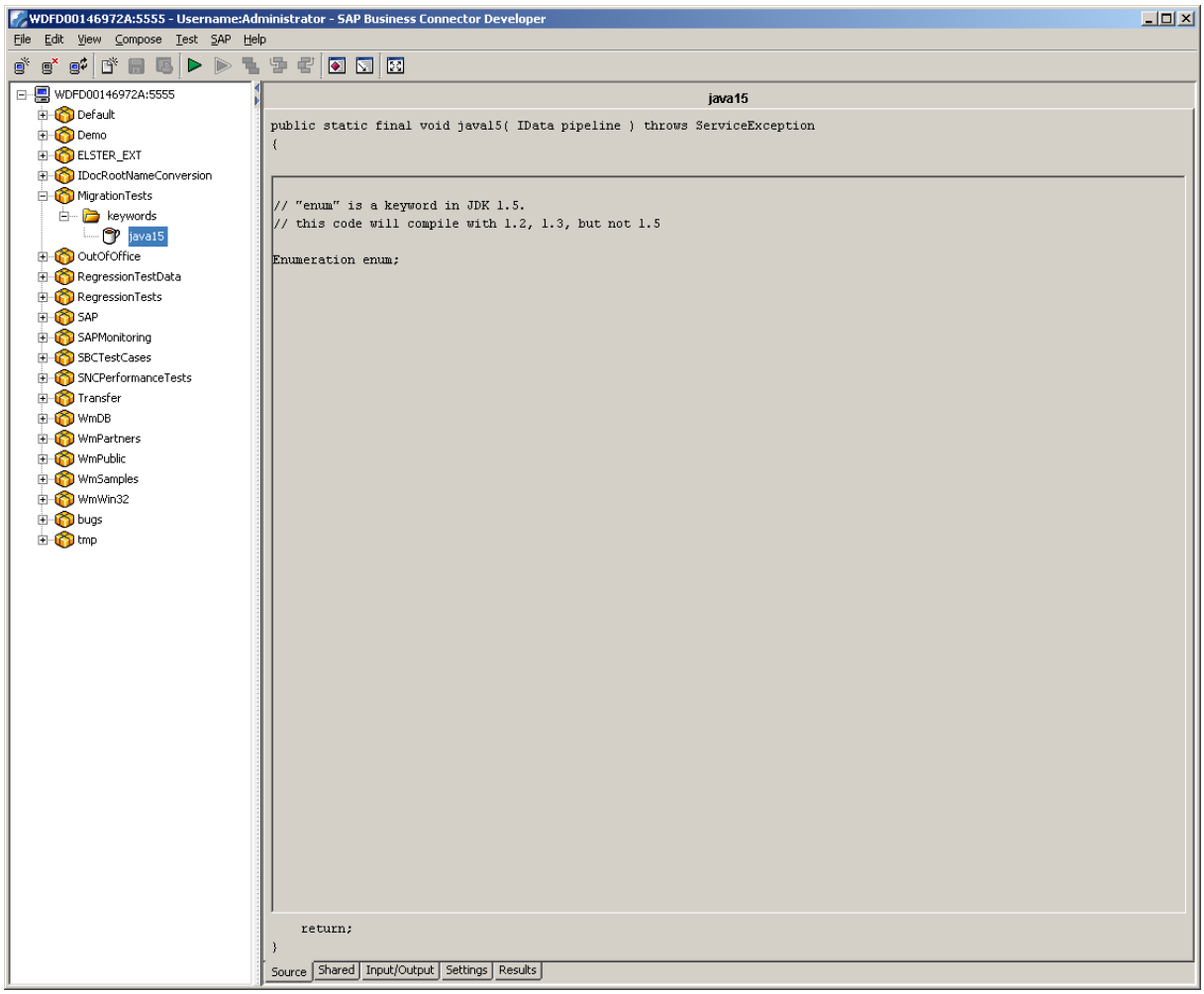
Remove Java compile time incompatibilities if you need to change the Java source

Although there are some Java language compile time incompatibilities between Java 1.2 and 1.5, you will not notice these issues until you need to make modifications to your Java code.

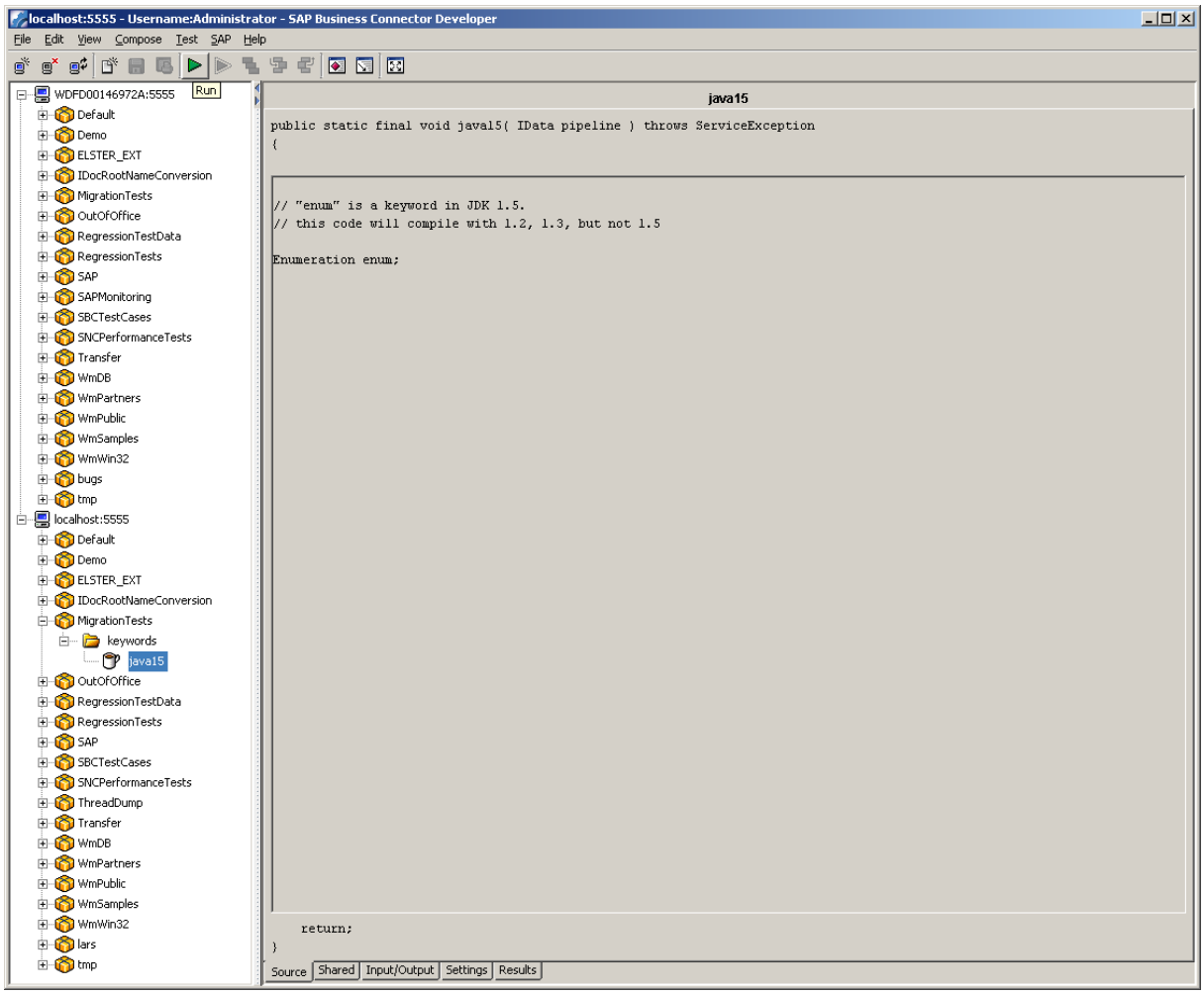
The following examples will make this clearer.

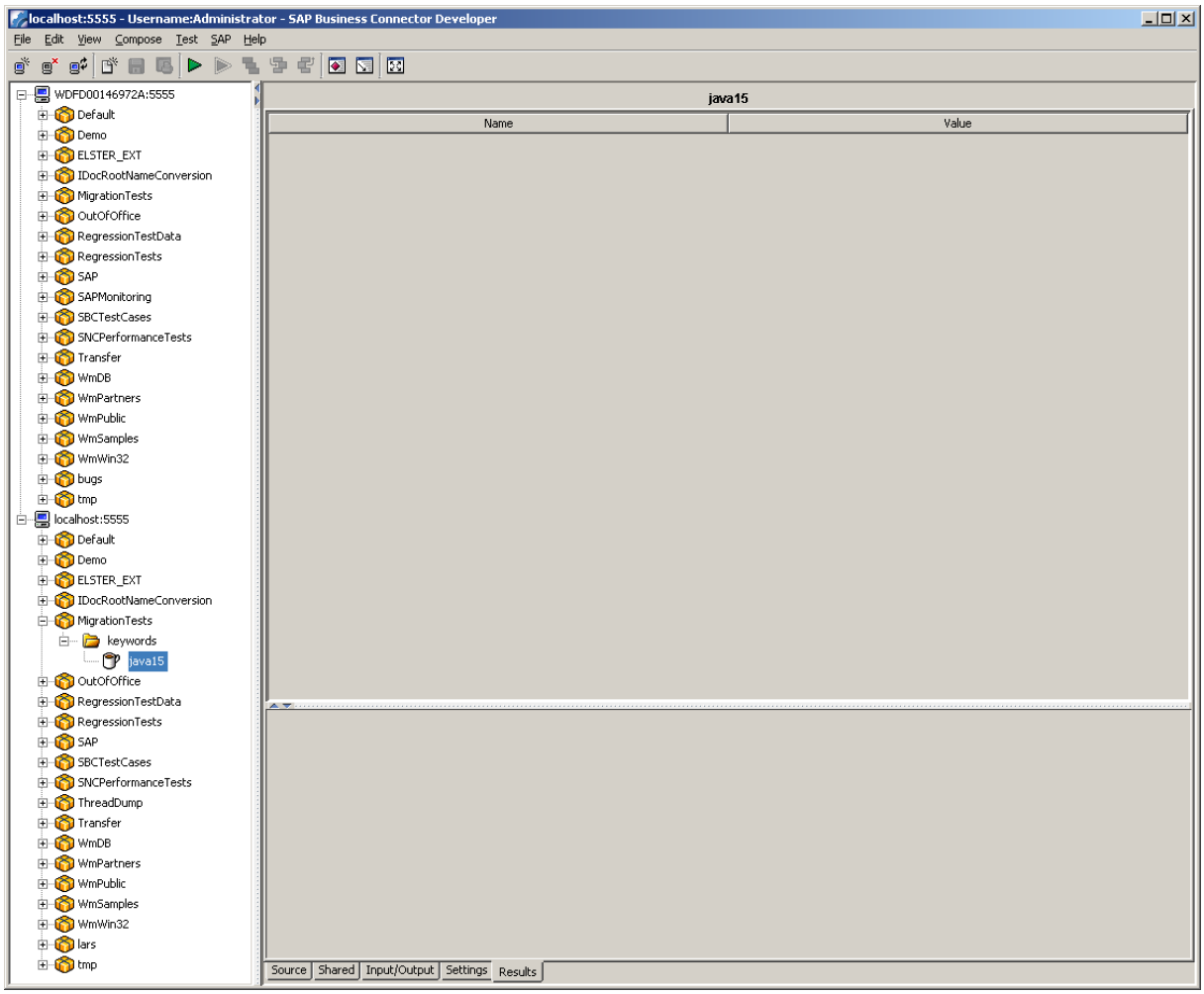
Enum keyword

Suppose you have created the following Java Service in BC 4.7:

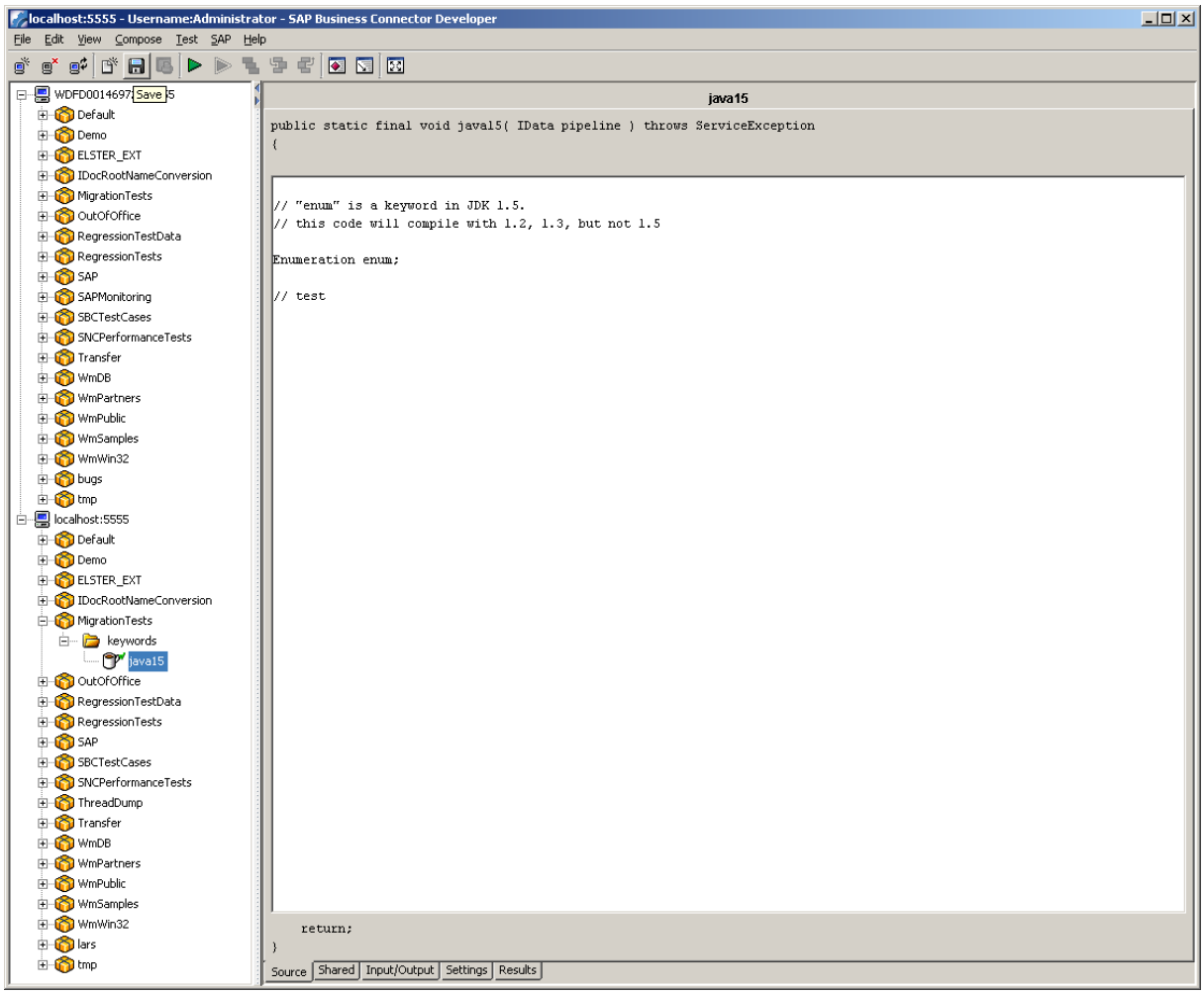


This code will compile in BC 4.7. You can also transfer this package without problems to BC 4.8 and run it on 4.8:

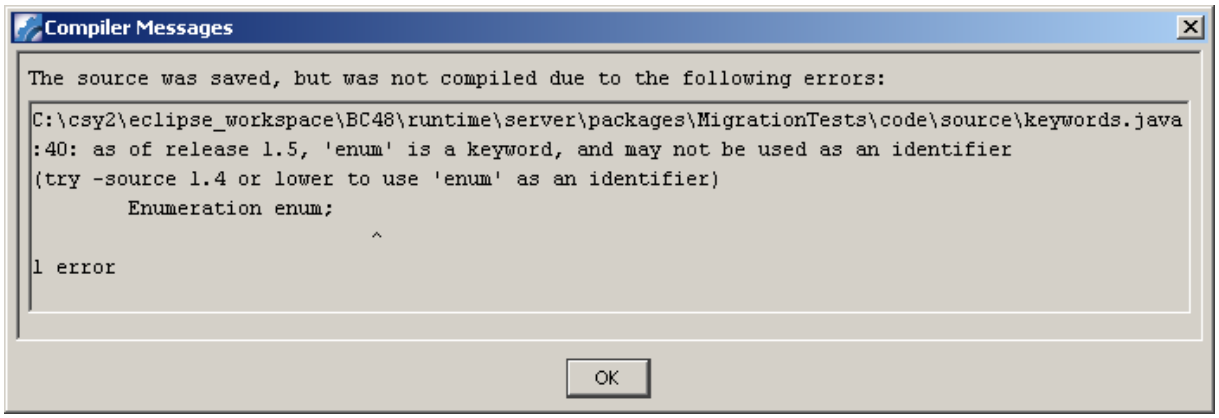

localhost:5555 - Username:Administrator - SAP Business Connector Developer
File Edit View Compose Test SAP Help
Run
WDFD00146972A:5555
Default
Demo
ELSTER_EXT
IDocRootNameConversion
MigrationTests
OutOfOffice
RegressionTestData
RegressionTests
SAP
SAPMonitoring
SBCTestCases
SNCPerformanceTests
Transfer
WmDB
WmPartners
WmPublic
WmSamples
WmWin32
bugs
tmp
localhost:5555
Default
Demo
ELSTER_EXT
IDocRootNameConversion
MigrationTests
keywords
java15
OutOfOffice
RegressionTestData
RegressionTests
SAP
SBCTestCases
SNCPerformanceTests
ThreadDump
Transfer
WmDB
WmPartners
WmPublic
WmSamples
WmWin32
lars
tmp
java15
public static final void java15(IData pipeline) throws ServiceException
{
// "enum" is a keyword in JDK 1.5.
// this code will compile with 1.2, 1.3, but not 1.5
Enumeration enum;
return;
}
Source Shared Input/Output Settings Results



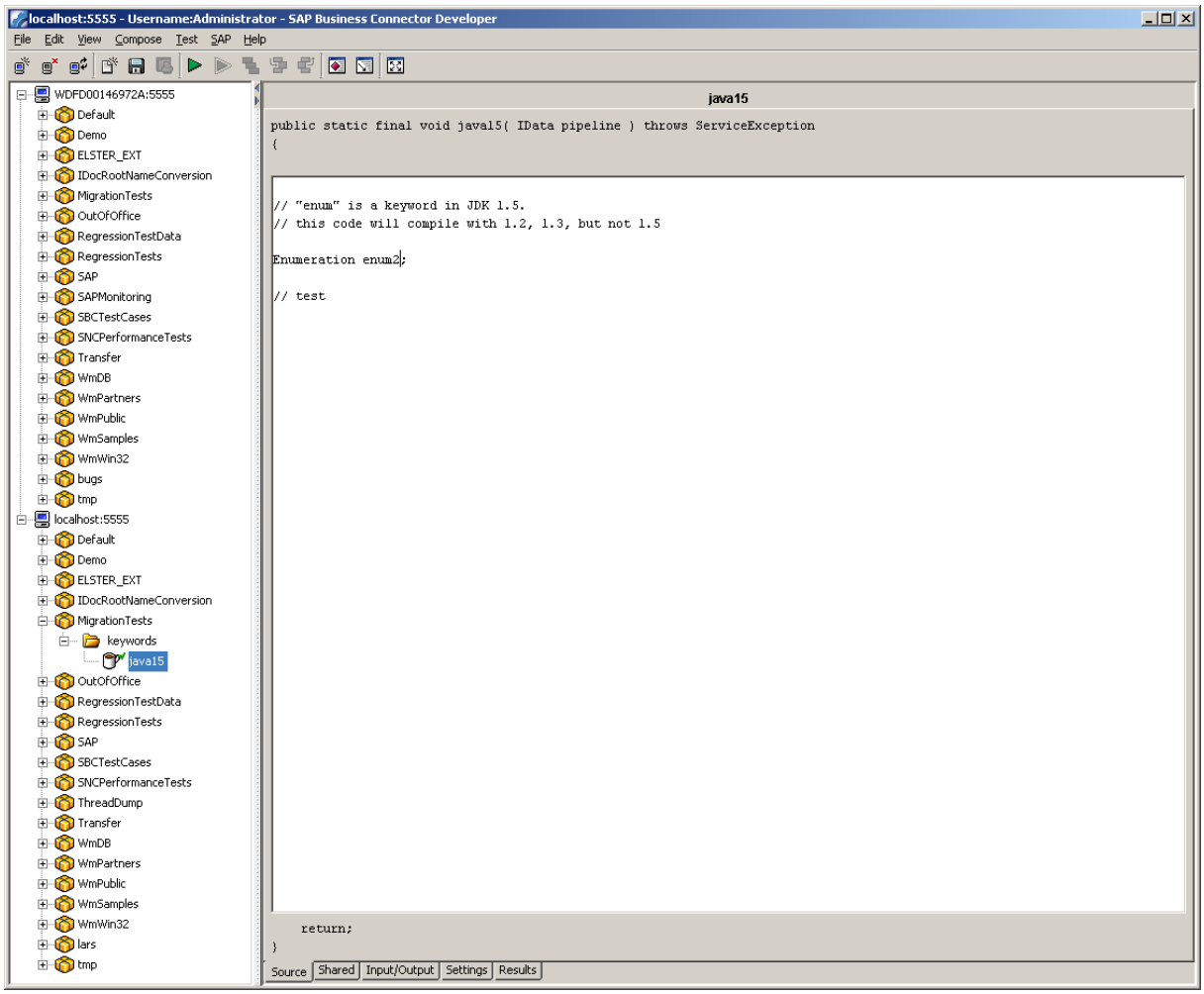
But, you cannot recompile this code without modifying it:



When saving, you will get a compiler error:

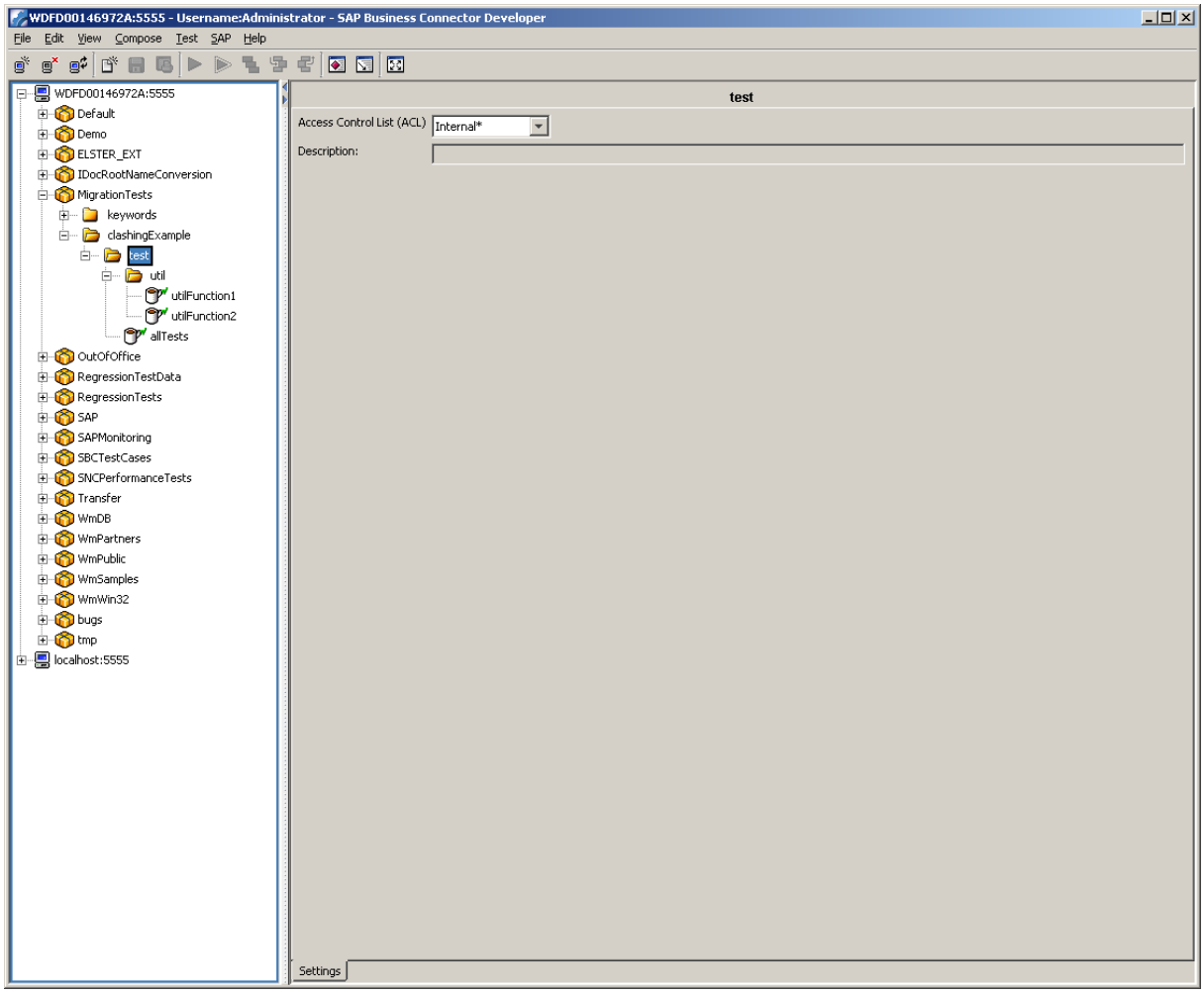


Fix the problem, then you can compile successfully:

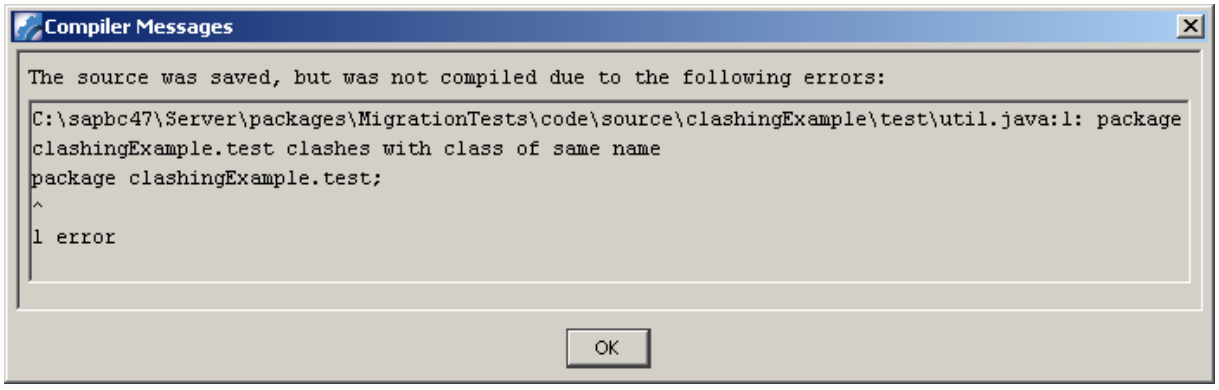

localhost:5555 - Username:Administrator - SAP Business Connector Developer
File Edit View Compose Test SAP Help
WDFD00146972A:5555
Default
Demo
ELSTER_EXT
IDocRootNameConversion
MigrationTests
OutOfOffice
RegressionTestData
RegressionTests
SAP
SAPMonitoring
SBCTestCases
SNCPerformanceTests
Transfer
WmDB
WmPartners
WmPublic
WmSamples
WmWin32
bugs
tmp
localhost:5555
Default
Demo
ELSTER_EXT
IDocRootNameConversion
MigrationTests
keywords
java15
OutOfOffice
RegressionTestData
RegressionTests
SAP
SBCTestCases
SNCPerformanceTests
ThreadDump
Transfer
WmDB
WmPartners
WmPublic
WmSamples
WmWin32
lars
tmp
java15
public static final void java15(IData pipeline) throws ServiceException
{
// "enum" is a keyword in JDK 1.5.
// this code will compile with 1.2, 1.3, but not 1.5
Enumeration enum2;
// test
return;
}
Source Shared Input/Output Settings Results

Package/class name clashing

The enum issue is one of the easier compile-time issues. A bigger issue is package/class name clashing. Suppose you have the following in BC 4.7:



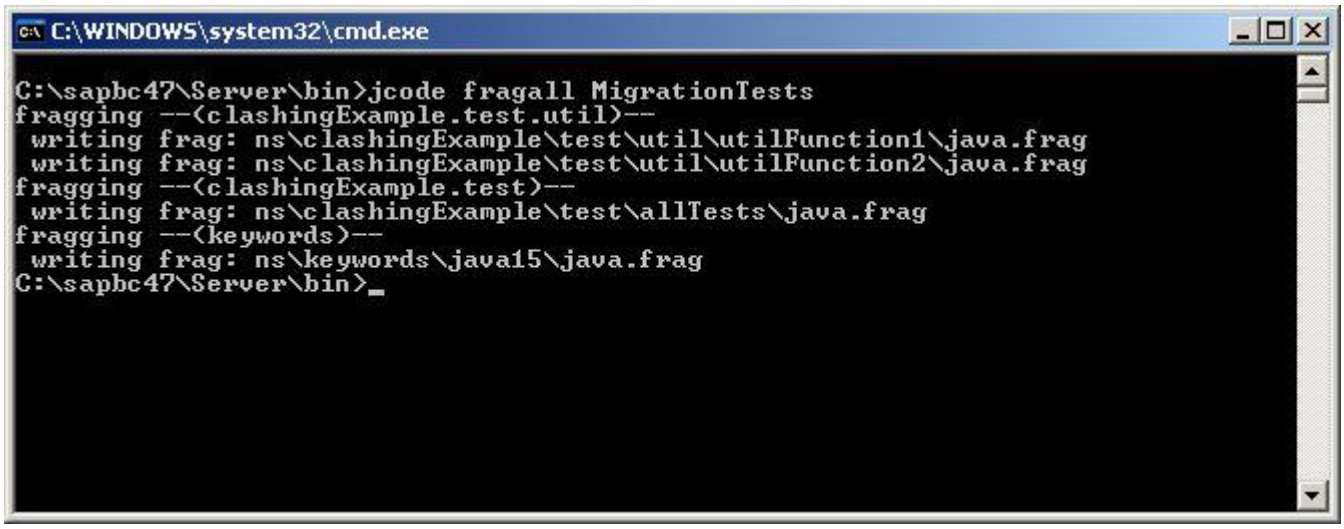
You have a folder test, in which you implement a Java Service “allTests”. And you have a sub-folder util, where you also have Java-Services (a quite common scenario). This structure was possible under JDK 1.2 compiler, but not anymore in later JDK versions. So when you try to recompile this Service in BC 4.8, you will get the following error:



In such a case, you need to have the source code for all Java Services in the concerned package, the source code must fully comply with the jcode template as described in the examples of appendix F in the SAP Business Connector Developer Guide, and you have to do the following steps:

a) goto your OLD BC, change to directory bin and run jcode fragall <your package name>

Example:



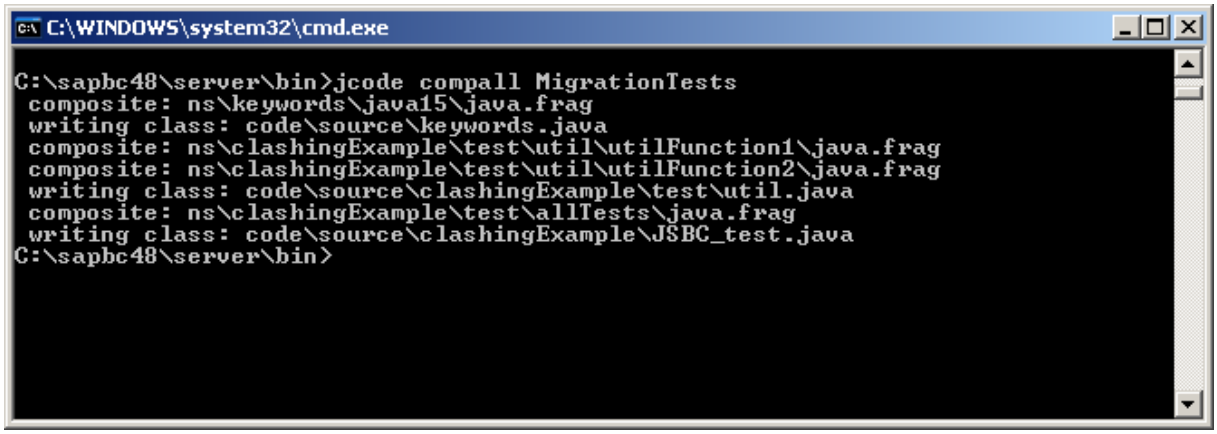
b) delete the `packages\<your package name>\ns` directory in the new BC

c) copy the directory `packages\<your package name>\ns` from your old BC into the new BC directory `packages\<your package name>`

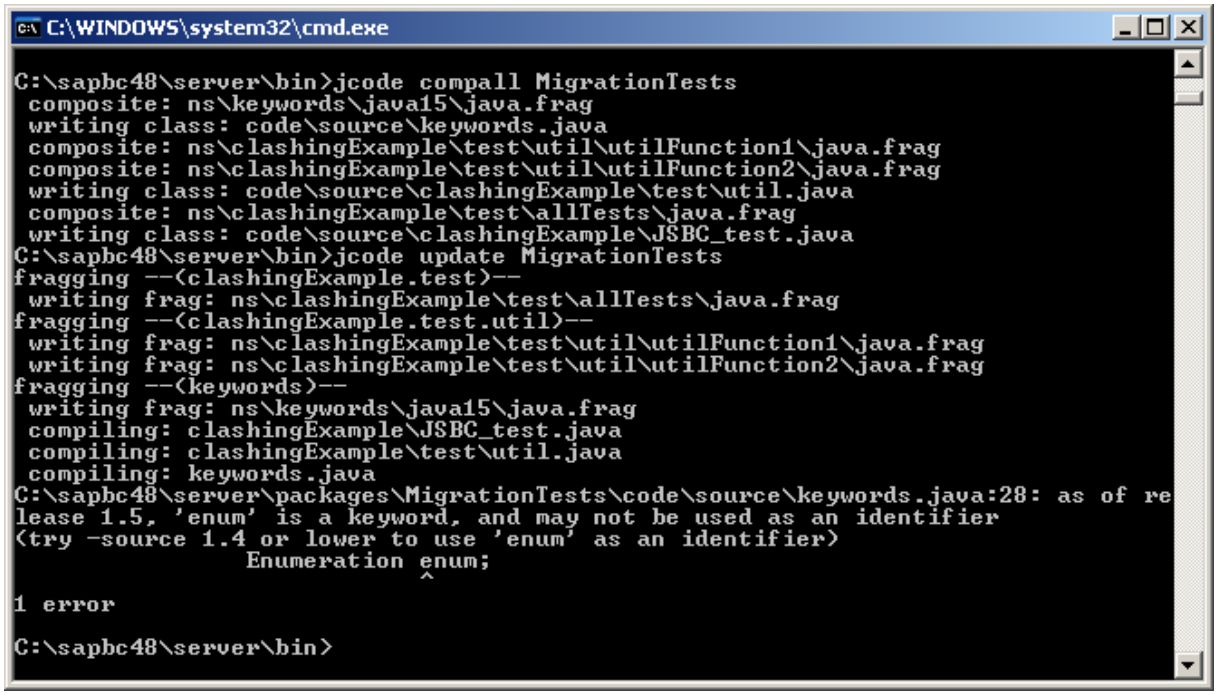
d) go to your new BC and delete all files under `packages\<your package name>\classes`

e) go to your new BC and delete all files under `packages\<your package name>\source`

f) change to directory `bin` and run the command
`jcode compall <your package name>`

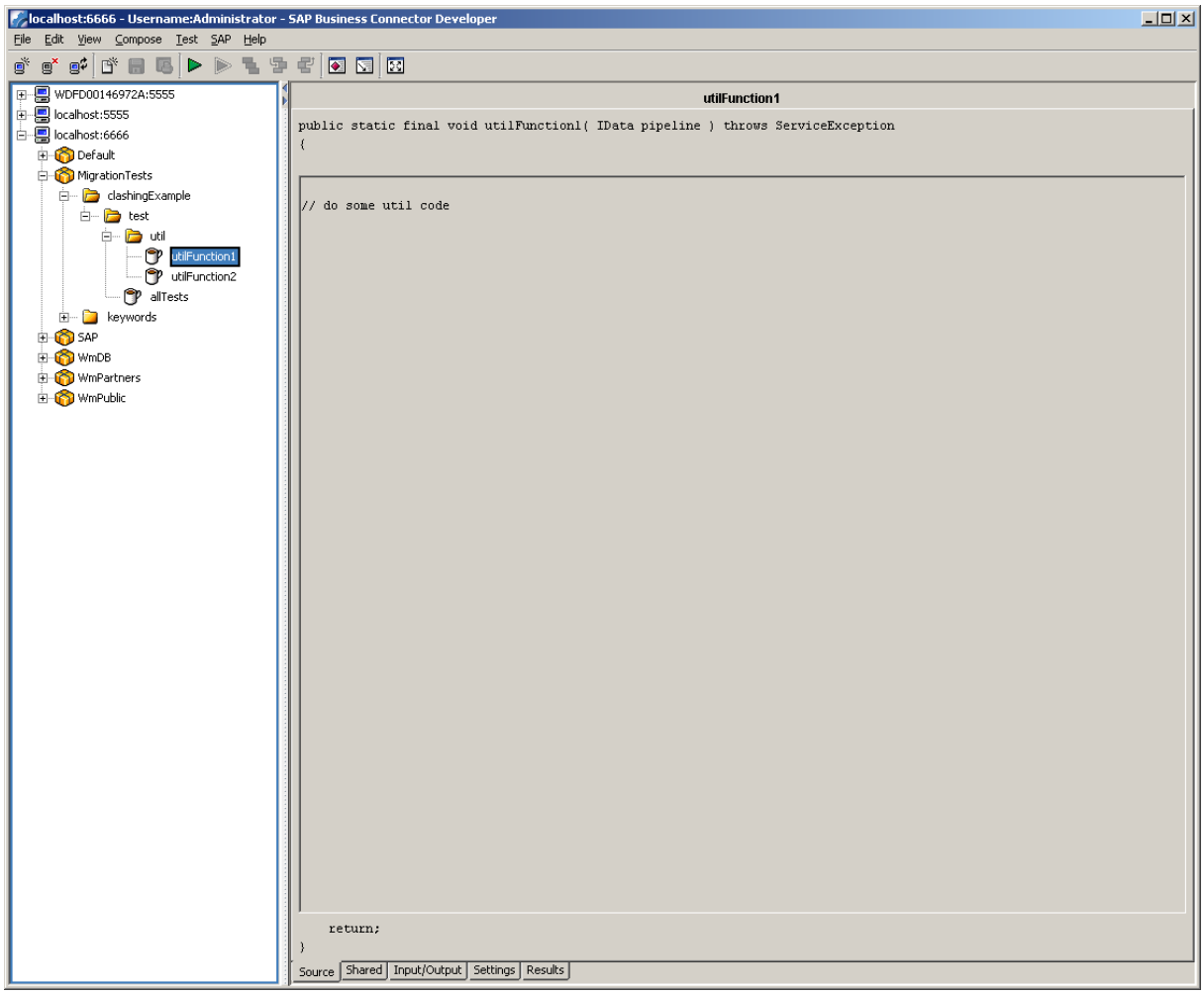


g) then run the command
`jcode update <your package name>`
in the same directory.



(The enum problem can then be fixed in the Developer as described above. This is not the topic here.)

Now you can start BC Developer (or refresh the package if already started):



Unlock the utilFunction1 and make a change. This will work now without error.

Missing WmDB dependencies

If you are using some of the DB API classes, you must add a dependency to the WmDB package version 2.0 in your package, otherwise the code will not compile anymore.

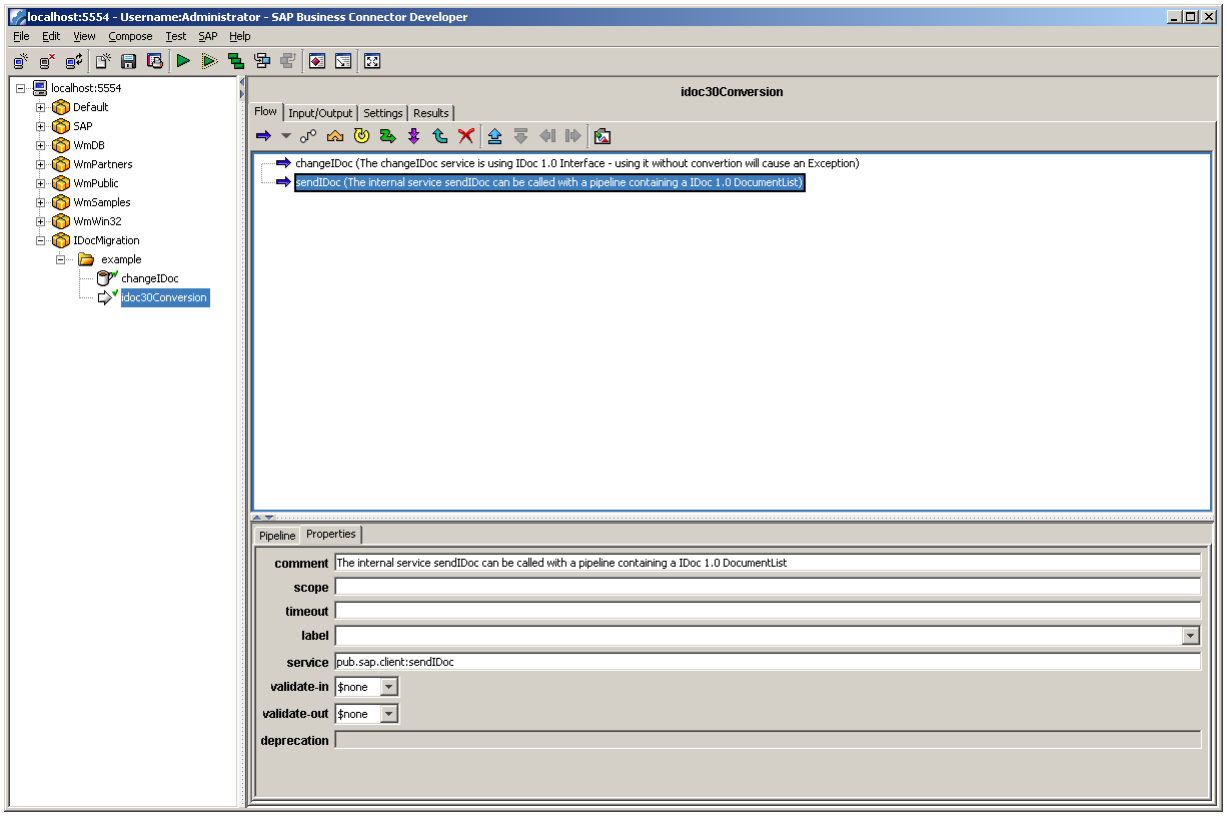
Migrating Services using the Java IDoc Class Library 1.0

As described before the Business Connector 4.8 uses the Java IDoc Class Library 3.0. Services using the Java IDoc Class Library 1.0 need to be adjusted. Business Connector 4.8 provides two solutions for this step.

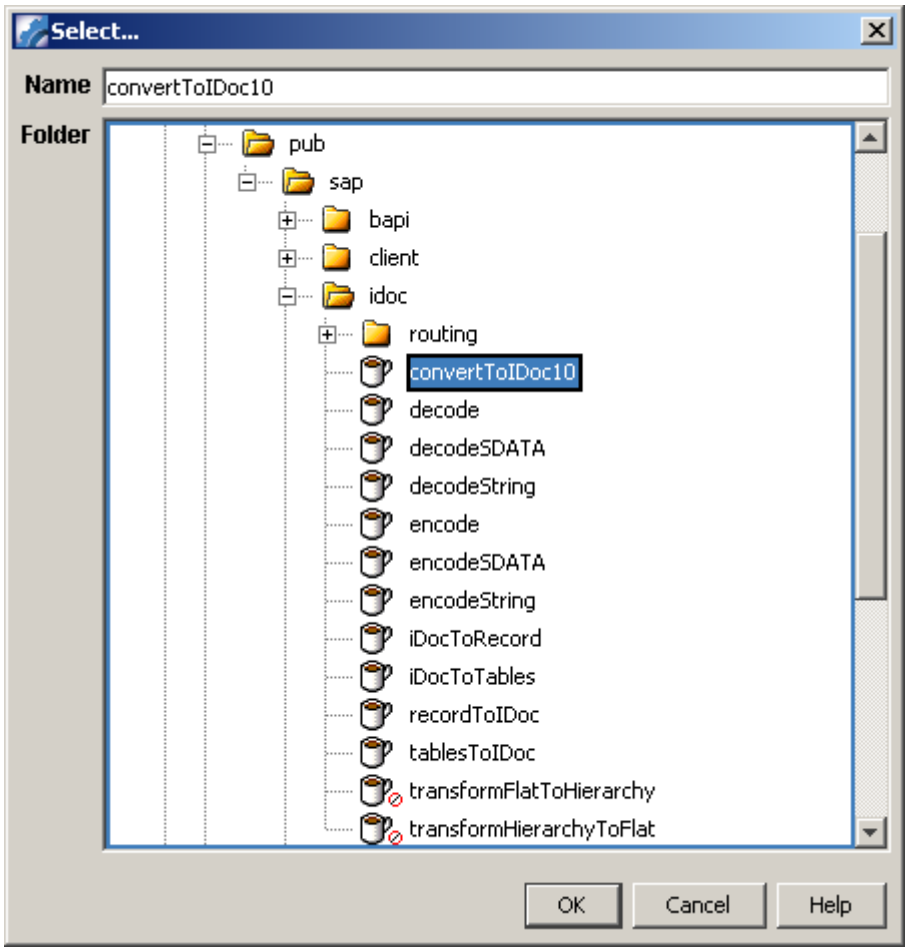
Including the Conversion Service

Internally all IDoc functionality is implemented using Java IDoc Library 3.0 classes, but all Services delivered with BC 4.8 are also IDoc 1.0 aware. Therefore you do not need to convert when calling such a built-in Service – the conversion will be done internally. Conversion is

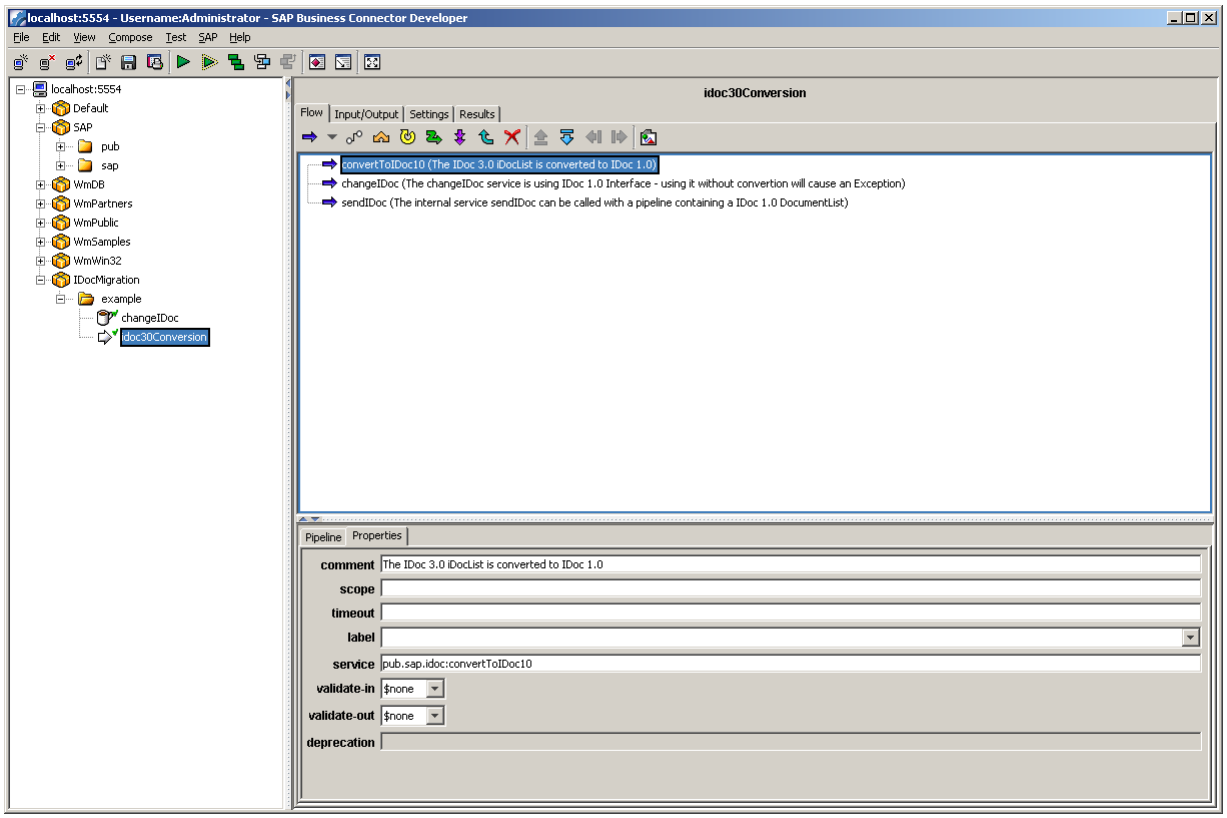
needed whenever you call a Service which is implemented using the IDoc 1.0 interface. The steps are shown in the following example:



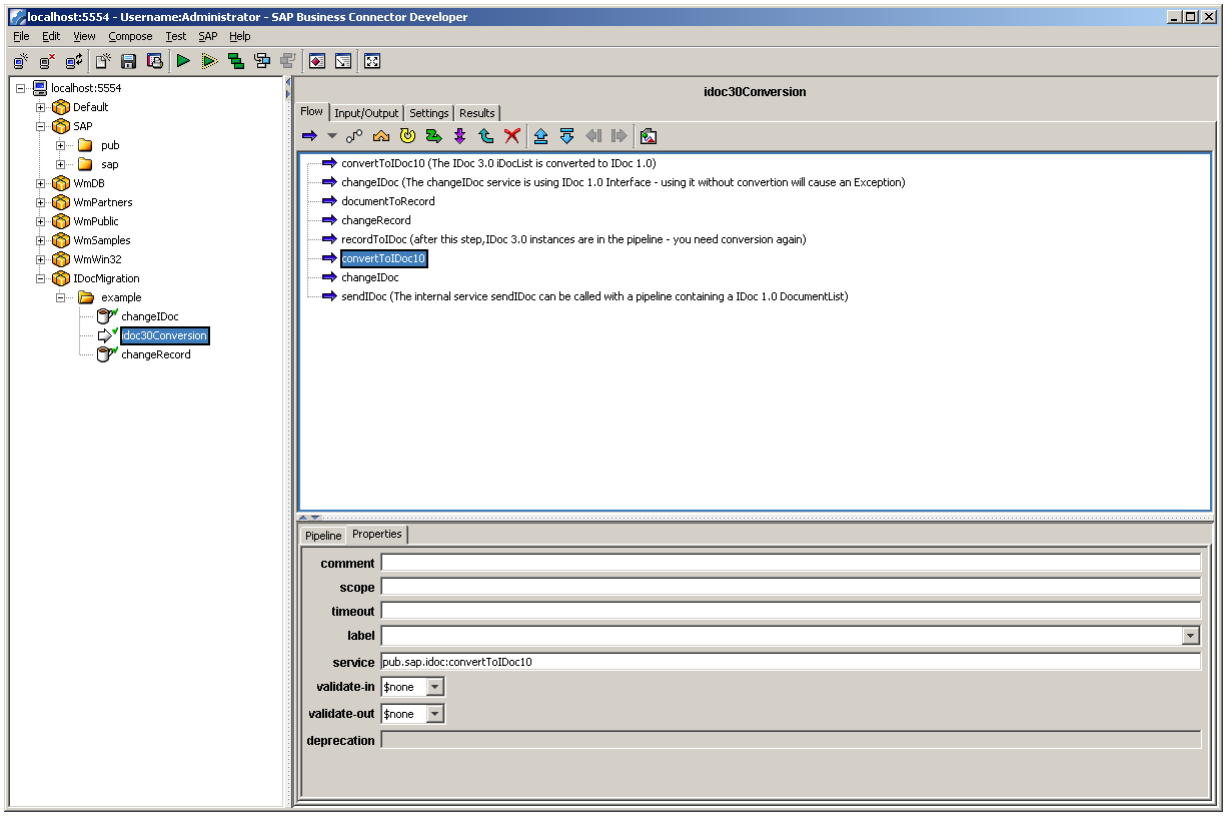
This example shows a simple flow including a Java Service using Java IDoc Class Library 1.0 (changeIDoc). The flow needs an iDocList as input parameter, changes this list in the changeIDoc Service and sends it. The input parameter will be an IDoc.DocumentList (IDoc 3.0), so you need to add the conversion Service before invoking the changeIDoc Service.



The conversion Service is located in pub.sap.idoc and is called convertToIDoc10. Add this Service to your flow and put it before your Service is invoked.



Save your flow. Any IDocList passed to the flow in the pipeline will be converted to IDoc Library 1.0 now. Be sure to add the conversion Service before any of your own Services using IDoc Library 1.0.



In this example you need to call the conversion Service 2 times.

Avoid the performance decreasing conversion whenever possible. Think about migrating your Services to Java IDoc Library 3.0 soon.

Migration of your Services to Java IDoc Library 3.0

To avoid the conversion step described in the previous section it is necessary to convert your Services to Java IDoc Class Library 3.0. Some of the steps you need to do can be automated, but you may need to do some individual coding. The following steps can be automated:

1) Classes are renamed. Inner classes or interfaces like IDoc.Document, IDoc.DocumentList etc. are renamed to IDocDocument, IDocDocumentList etc.
2) Old imports are removed, the new ones added
3) Method names and signatures are changed to fit to the new interface. This is done for static methods and for all methods called for an object stored in a variable.

In any case you should review the changes made by the conversion tool. Handcrafting is needed for the following steps:

1) Method cascades will not be changed. Structures like

   document.getRootSegment().getFirstChild().getNext("name").setField("key", "value");

   will not be handled.
2) Exception handling conversion cannot be automated in every case. The exceptions thrown in the Java IDoc class Library 3.0 implementation need to be handled.
3) XML creation methods have been removed from the Interfaces in IDoc 3.0 implementation. You need to use the Service pub.sap.idoc:encode instead.
4) The class IDoc.NestedException has been removed. It is not needed any more, because of the cause throwable feature in Java 5.

These topics are covered in more detail in the IDocLibraryMigrationGuide_10_30.

The tool for automatic conversion is included in the the jcode utility, which is part of the Business Connector 4.8 installation. Migration is done by calling

```
jcode migrate <pkg>
```

or

```
jcode migrate <pkg> <name>.
```

The packages to migrate need to be migrated to a Business Connector 4.8 installation as described before. No backup will be done of the java files, so ensure to do backup before starting the migration. Do a **`jcode frag <pkg> <name>`** or **`jcode fragall <pkg>`** afterwards to be able to edit the Services in the Developer again.

The migration process may print out some warnings to the console which may help you during your manual rework.

Example:

If you want to migrate a Service named *pub.idoc:fillidoc* in a package called *IDocMigration* you do the following:

```
jcode migrate IDocMigration pub.idoc:fillidoc
```

This will run over the java file associated to the Service and do the automatic replacements there. After the replacement is done, you need to split the java file to fragments again to be able to edit the Services in the developer. To do this call

```
jcode frag IDocMigration pub.idoc:fillidoc
```

You will now see the changes when editing the Service in the developer. Take a look at the result and do your manual changes there.

If you want to migrate the whole package (named IDocMigration) do the following:

```
jcode migrate IDocMigration
```

and

```
jcode frag IDocMigration
```

Please be patient. Due to some complex algorithms the migration of a whole package may take some time.

For example java source demonstrating IDoc/Segment parsing with the new IDoc 3.0 library see packages/SAP/code/source/sap/demo/idoc/mappings.java and the sections “Accessing and Modifying Fields in IDocs”, “Constructing an IDoc with the SAP Java IDoc Library” in the SAPBCSapAdapterGuide.

You also should replace the usage of the following classes with their new counterpart:

Old	**New**
BCDocument	BCIDocDocument
BCDocumentList	BCIDocDocumentList
BCRecord	BCIDocRecord
BCSegment	BCIDocSegment
IDOC	Don’t use anymore
IDOCControl	Don’t use anymore
IDOCSegment	Don’t use anymore
JCoBCDocumentList	JCoBCIDocDocumentList

Migrating Services using WmDB package

The WmDB package was redesigned in 4.8. The main goal was to make this package more robust against failures. It provides the following enhancements:

- Implicit connection management, meaning you do not have to explicitly open and close the connection anymore. Instead, you just use the DB Service you need (e.g. insert, query, execSQL, etc.) and the Service will automatically take a connection from the pool and return it after it is done. When using multi-step transactions, the startTransaction Service automatically stores the connection in the current B2B Session for the following DB Services to use. When you end the transaction (clearTransaction, rollback or commit), the connection is released again. Also when a failure occurs in a DB-operation, the connection is released.
- Distinction between global pool and session-based pool was removed. There is only a global pool now.
- Connection sweeper for connections with killed sessions. (I.e. a startTransaction was never closed by a corresponding commit/rollback.)
- You can now enter a testsql statement for every DB alias. This statement is then used to check in regular times if the connection is still valid.
- You can optionally fallback to the old behaviour if you need to. In this case, please add the following to your Extended Settings:
  `watt.server.db.compatibilityMode=true`
  Be aware that this setting may not be available anymore in a future BC release.