Document Version: 1.0 – 2016-01-20

CUSTOMER

JCo 3.0 in Web Channel 7.54

Configuration & Migration Help

Typographic Conventions

Type Style	Description
Example	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Textual cross-references to other documents.
Example	Emphasized words or expressions.
`EXAMPLE`	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`	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`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE`	Keys on the keyboard, for example, `F2` or `ENTER`.

Document History

Version	Date	Change
1.0	March 2016	Initial Version

Contents

1 Introduction 5

2 Destination Configuration 8
2.1 Setting Connection Properties for WEC 8
2.2 Naming Conventions for Destinations 13
2.3 Destination Generation in XCM 14
2.4 Destination Configuration for Product Configuration Scenarios 17
2.4.1 Integration within CRM: SSO Authentication 17
2.4.2 Integration within CRM: Authentication with Technical User 20
2.4.3 Integration within ERP: Authentication with Technical User 20

3 Migration Steps from JCo 2.x to JCo 3.0 21
3.1 Package and Naming Changes 21
3.1.1 Change DC dependencies 21
3.1.2 Resolve Build Errors 22
3.2 JCo Migration Helper 24

4 Troubleshooting 25

1 Introduction

This document introduces the Connection Handling related changes in SAP CRM Web Channel/SAP ERP E-Commerce of Version 7.54.

This introductory section aims to create a basic understanding of terms and technologies, followed by three main parts:

1. New setup steps to establish a valid backend connection

   → WEC has a new way to specify connection details (now destinations). For IPC additional considerations are required.

   → Additional case specific destinations can be generated.

   → Naming Conventions **must** be considered when creating destinations manually.
2. Migration steps

   → Adaptations are required in customer coding in order to support the new connection handling procedure.
3. Troubleshooting

   → Typical Errors and how to solve them

This guide does NOT intend to provide detailed information about connection handling with WEC in general. The focus is set on differences compared to older versions (i.e. versions 5.0 up to 7.33).

SAP CRM Web Channel/SAP ERP E-Commerce Version 7.54

> **Note**
>
> The solution *SAP CRM Web Channel 7.54* is delivered with *SAP CRM 7.0 EHP4*.
>
> The solution *SAP ERP E-Commerce 7.54* is delivered with *SAP ERP 6.0 EHP8*.
>
> For simplicity reasons both are referred to in the following as **WEC 7.54** or **WEC**.

Please refer to note 1583830 for further release and maintenance cycle details.

WEC 7.54 is functional equivalent to its predecessor version WEC 7.33.

This means from a business perspective, both versions offer the same functionality.

However, some **technical changes** have been applied to WEC 7.54:

1. WEC 7.54 runs on NetWeaver 7.5, which is running with Java 8.
2. For parsing XML files, the SAP XMLToolkit is no longer used. Instead, Java XML Parsing is facilitated.

3. The connection handling between Java and ABAP systems has been adopted to the latest version of SAP Java Connector.

SAP Java Connector (JCo)

The SAP Java Connector (SAP JCo) is a toolkit that allows a Java application to communicate with any SAP system. It combines an easy to use API with unprecedented flexibility and performance. The package supports both, Java to SAP System as well as SAP System to Java calls.

SAP JCo is available in two different environments: A standalone version and one integrated into the SAP NetWeaver Application Server Java (short: NW). Both variants are based on a compatible API.

> **Note**
>
> The latest standalone version currently is JCo 3.0, which is significantly different from its predecessors JCo 2.0 and JCo 2.1.
>
> More information on the release strategy are documented in notes: 549268 and 1077727

WEC and SAP JCo Versions

Different WEC versions are based on different versions of the JCo API for connecting to CRM/ ERP backend systems.

WEC Version	Based on JCo API Version
5.0, 7.0, 7.01, 7.02 (based on NW 7.0x)	**JCo 2.0**
7.30, 7.31, 7.32, 7.33 (based on NW 7.3x, NW7.4)	**JCo 2.1**
7.54 (based on NW 7.5)	**JCo 3.0 via Destination Service**

WEC is using the JCo variant that is integrated into NW. However, in WEC 7.54 the JCo API is not directly used for managing the backend connections. Instead, we rely on the destination service as part of NW.

Destination Service

The destination service is integrated into SAP NetWeaver and offers a central service for managing client side information required to connect to (backend) services. As of now, the destination service supports HTTP and RFC connectivity.

Each destination represents one connectivity end-point at a target system. The destination service provides a secure storage for the connection properties.

The properties of the destinations can be changed programmatically via the destination service API, but also manually via the destination service UI that is integrated in the NetWeaver Administrator.

See Destination Service Help *for more information.*

2 Destination Configuration

This chapter targets all users who are maintaining backend connection details for WEC applications with version 7.54.

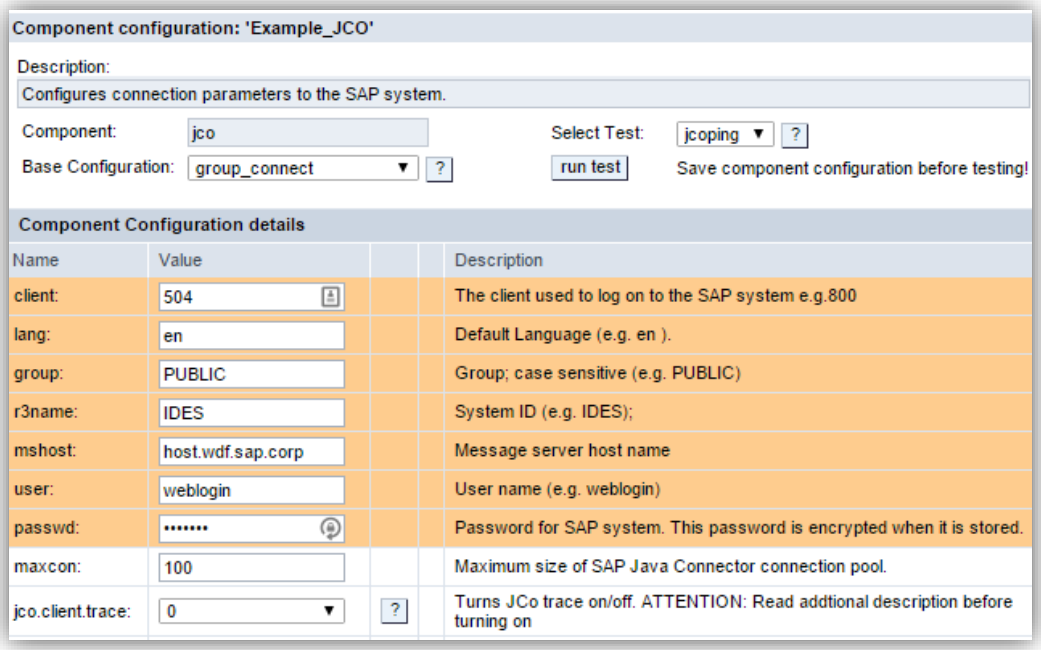
Important

As of version 7.54, the connection properties to the backend systems are no longer maintained within the Extended Configuration Management (XCM) of the web application, but only referenced there. Instead, the Destination Service offered by NetWeaver is used to specify the connection details.

2.1 Setting Connection Properties for WEC

Old maintenance with XCM

In prior versions, connection properties have been maintained in XCM:



Old XCM UI to Maintain Backend Connection Properties

For more information about the maintenance of JCO properties in XCM (WEC version lower than 7.54), please have a look at *Development & Extension Guide for Web Channel on NW7.0/7.3*.

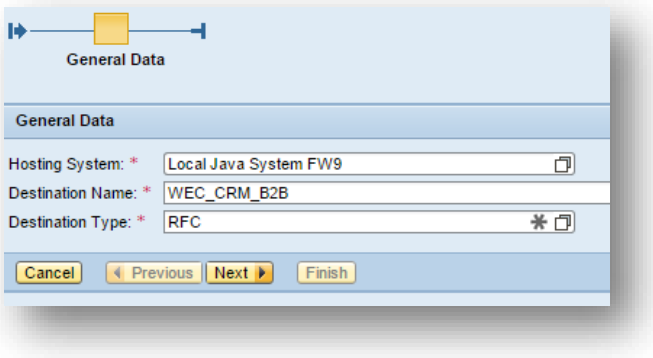
New Maintenance with Destination Service

Connection Properties for one specific backend are maintained as so called *destinations* within Destination Service accessible via the SAP NetWeaver Administrator (short: NWA).

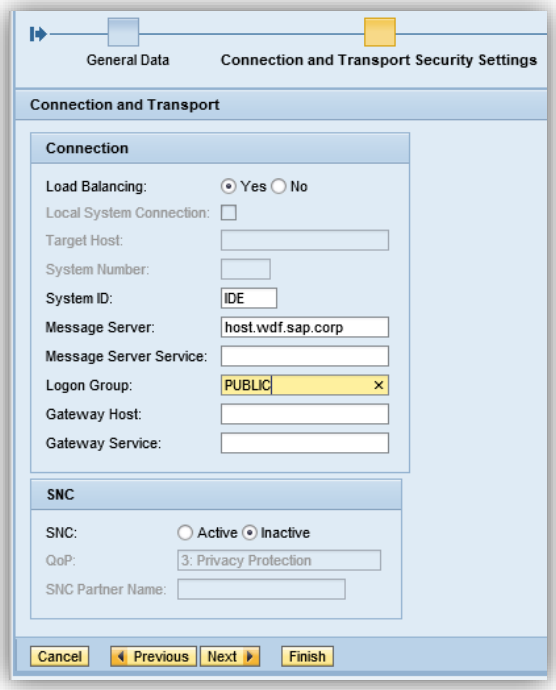
> **Note**
>
> You can find the Destination service in the SAP NetWeaver Administrator via:
>
> **`https://<host>:<port>/nwa`**
> → *System Administration* →*Configuration* → *Destinations*
> or the direct link
> **`https://<host>:<port>/webdynpro/resources/sap.com/tc~lm~itsam~ui~mainframe~wd/FloorPlanApp?applicationID=com.sap.itsam.cfg.sec.destinations&applicationViewID=destinations`**

The destination properties for connecting to ABAP systems have to be of type **RFC Destination.**

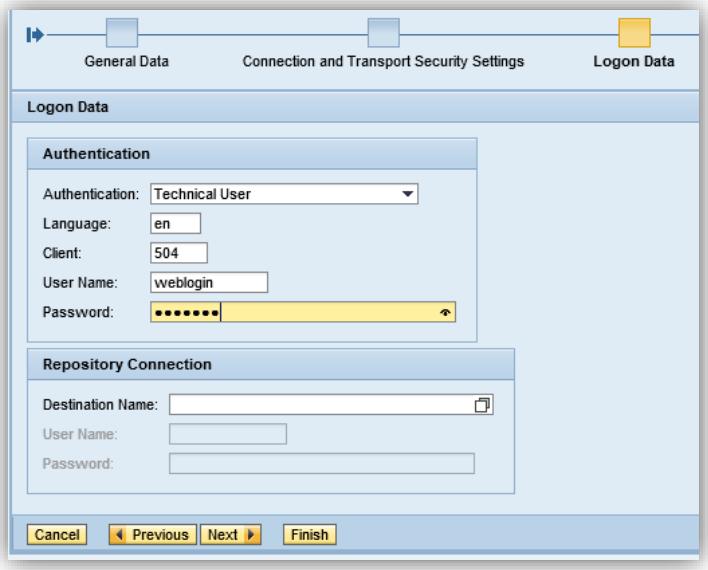
By selecting the *Create...* Button, a wizard opens that guides through the destination setup process:



Create a new RFC Destination - Step 1



Create a new RFC Destination - Step 2

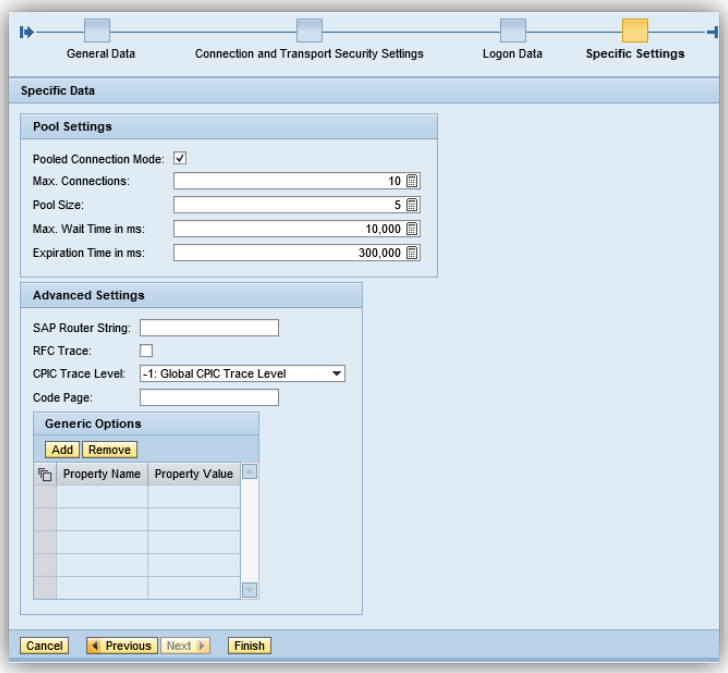


Create a new RFC Destination - Step 3

> **Note**
>
> The demonstrated example covers the Authentication Type *Technical User*. Further authentication types will be explained below.

Pooling Settings

Extended Pooling Properties can be defined in a last step of the destination creation process or later:



Create a new Destination - Pooling Options

In WEC pooling behavior can be configured for each connection definition in EAI-CONFIG. By default, pooling is switched on, however, it can be turned off with one of the following parameters:

```
<param name="destination.nopooling" value="true"/>
<param name=" jco.destination.pool_capacity" value="0"/>
```

> **Pooling information is only relevant for Stateless[1] Connections!**
>
> When pooling is turned off, the stateless connection instances will be destroyed and rebuilt for each function call.

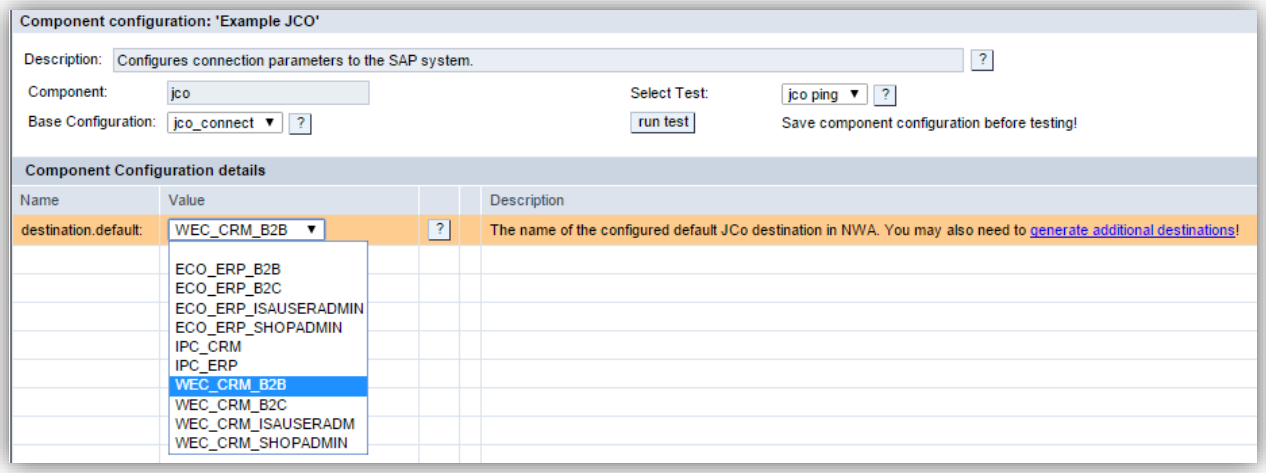
[1] A Stateless Connection is the default connection type. A connection pool (configurable) holds a pre-defined number of stateless connections. Those connections will be allocated for exactly one function call without a special context and returned to the pool afterwards.

Stateful Connections, however, get a context assigned on initialization. As long as this context is assigned (bound to the session lifetime), the connections will not be returned to the pool, no matter if pooling is configured or not.

Mapping a Destination to a Specific XCM Scenario

The destination - created in NWA - has to be assigned to a specific WEC scenario configuration to tell the application which backend connection shall be used. The assignment is done within XCM for each application scenario within component *jco*.

After creating a new customer jco component a screen like below is displayed:



New XCM UI: Choose a Destination

The parameter *destination.default* offers a dropdown box containing a list of all destinations defined within NWA.

The Component Test *jco ping* (started via *run test*) provides feedback to the user whether a ping with the selected destination is successful or not. This functionality is similar to the *Ping Destination* feature within NWA.

Recommendation

The description field offers a link generate additional destinations where destinations can be generated! It is highly recommended to use this for creating additional destinations. More details are provided in chapter 2.3.

Connection Authentication via Destination Service

The **WEC** application supports different **procedures on how to establish an authenticated connection** with the backend system.

- Connection with a user/password that has been maintained in the backend system
- Connection via NW login procedures (client certificate, user logon with own user store or LDAP)
- Connection with SSO cookie

> **Note**
>
> Detailed information on the supported login procedures for WEC is described in note: 1929623

The **destination service offers different authentication types** for connecting to the target system, whereby only the following are relevant for WEC:

- BASIC credential authentication - with a service user name and password
  => *Technical User*
- SAP Assertion Ticket - for the user of the current session accessing that destination
  => *Current User (Assertion Ticket)* or *Current User (Logon Ticket)*

See Destination Service Help *for more information.*

> **Follow these rules when creating new destinations for WEC:**
>
> 1. If WEC application should use **NW-based authentication** (e.g. UME, LDAP) you should use *Assertion Ticket*
> 2. When using **IPC application within product configuration scenario**, have a look at the chapter 2.4
> 3. Use *Technical User* for any other authentication type

2.2 Naming Conventions for Destinations

Within XCM only a so-called *base destination* can be selected. Depending on the configuration, like user login type, the actual destination will be determined at runtime. The **correct determination of destinations depends on naming conventions**, which are outlined below:

1. Authentication Type

Use Case	Full Destination Name
Authentication with **Technical User**	<baseDestination> *e.g. WEC_CRM_B2B*
Authentication with **UME**	<baseDestination>**_UME** *e.g. WEC_CRM_B2B_UME*
Autehntication with **SSO ticket**	<baseDestination>**_SSO** *e.g. IPC_ERP_SSO*

2. Connection Mode

Use Case	Full Destination Name
Load balancing	<baseDestination><?authentication> *e.g. WEC_CRM_B2B_UME*

Use Case	Full Destination Name
Direct server connection	<baseDestination><?authentication>**_DIRECT_<host>_<sysNo>** *e.g. IPC_ERP_SSO_DIRECT_host1_11*

3. Pooling Mode

As described earlier, pooling can be switched off for connections by setting one of the parameters: `destination.nopooling` OR `jco.destination.pool_capacity`

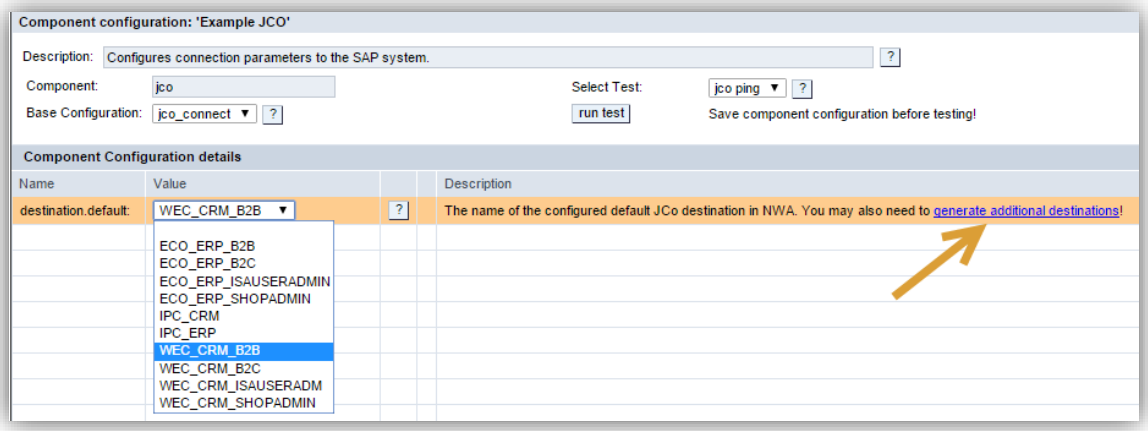
Use Case	Full Destination Name
Pooling Mode ON	<baseDestination><?authentication><?directConnection> *e.g. WEC_CRM_B2B_UME*
Pooling Mode OFF and load balancing	<baseDestination><?authentication>**_NOPOOL** *e.g. WEC_CRM_B2B_UME_NOPOOL*
Pooling Mode OFF and Direct server connection	<baseDestination><?authentication>**_DIRECT_NOPOOL_<host>_<sysNo>** *e.g. IPC_ERP_SSO_DIRECT_NOPOOL_host1_11*

2.3 Destination Generation in XCM

A new feature is offered, with which destinations can be generated following the prior described naming conventions.

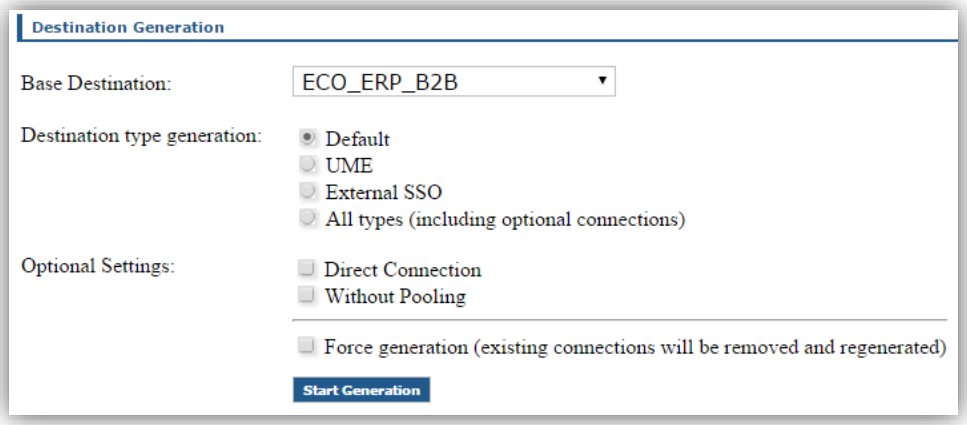
> **Note**
>
> Not all WEC scenarios require destinations in addition to the base destination with pre-configured technical user. You should consider generating destinations, in case:
>
> - You are using UME based authentication
> - You are using SSO tickets for authentication
> - You configured connections in EAI Config for which pooling is switched off
> - You want to connect to a specific application server instead of using load balancing

The **generation page** can be accessed via link generate additional destinations in the **JCO Component in XCM**:



XCM Access to Destination Generation

The following page opens:



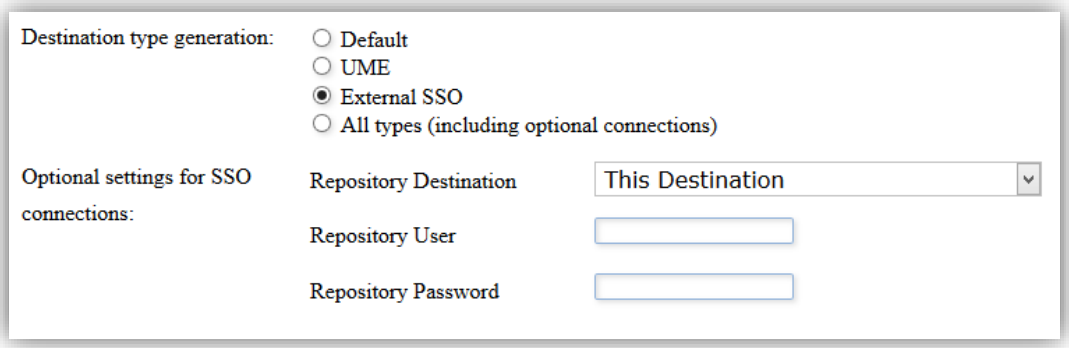
Destination Generation: Options

Features:

- The B*ase Destination* is taken as *template (by default the first one is selected)*
  - ⚠ The base destination must exist and has to be valid (connectable to the backend system)!
- *UME* and *SSO* destinations can be generated in addition
- Optional Settings allow to create multiple destinations with specific *pooling and/or connection modes*
- Existing destinations can be overwritten/regenerated (*Force generation*)
- When selecting option *Direct Connection* the application is retrieving a list of available application servers (based on the selected base destination) and generates new destinations for each of them
  - ⚠ The user defined for base destination needs authorizations to access function module `TH_SERVER_LIST` in order to retrieve the application server list
- Destination type *All Types* generates destinations for **ALL configuration combinations**

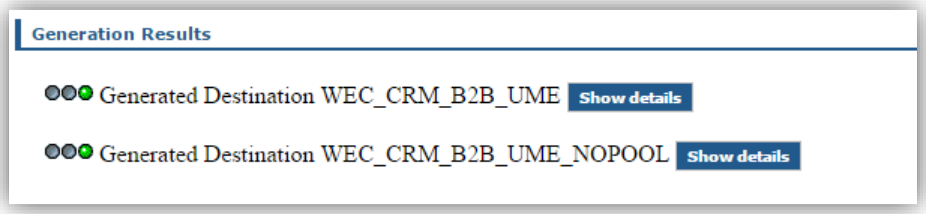
- For *SSO* and *All Types* selection, additional settings appear for specifying the repository connection (see chapter 2.4 for more details)

  → Repository settings will only be added if respective input fields are changed



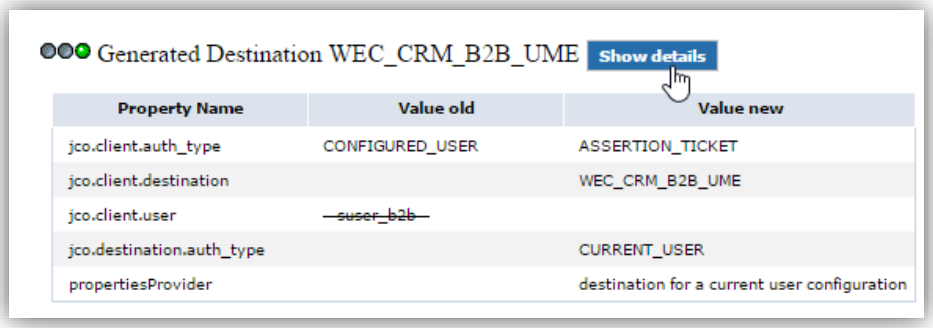
Destination Generation: Repository Destination

After selecting the base destination and appropriate options and clicking on *Start Generation*, a result list appears listing all generated destinations, for example:



Destination Generation: Result List

Clicking on the *Show Details* button, displays a table of the changes (compared to the base destination):



Destination Generation: Changed Destination Details

Caution

Generated destinations are not listed in XCM JCO component for selection. Only the base destinations can be selected there. The determination of the correction destination is handled internally.

2.4 Destination Configuration for Product Configuration Scenarios

This chapter targets users who are working with product configuration scenarios integrated into other applications/industry solutions, e.g. CRM Order

As part of the WEC solution, the crm/ipc application is offered. This application covers product configuration with integrated pricing in several scenarios. It is running on a NetWeaver AS Java similar to other WEC applications, but is usually called from within other systems or applications. Product Configuration can also be called standalone in ERP.

The product configuration UI is integrated in different applications and industry solutions:

- CRM Order
- ERP SD Order
- Vehicle Management
- Utility (via CRM Order)
- Telco (via CRM Order)

> **Important**
>
> If the **product configuration UI is called from a different scenario**, some IPC **specific considerations and appropriate actions** are necessary.
>
> If the crm/ipc application is called as integrated application from WEC, the connection handling from WEC applies.

An **authenticated session** is required for transferring data between backend systems (i.e. CRM) and the IPC web application.

→ When product configuration functionality is called from/within CRM, the SSO ticket is used to authenticate the user.

→ For integration in ERP sales process scenarios, the technical user shall be used for the RFC calls to the Configuration and Pricing Engines.

The following sections describe the specific preparation steps in the connection handling for the two different ways of authentication in more detail.

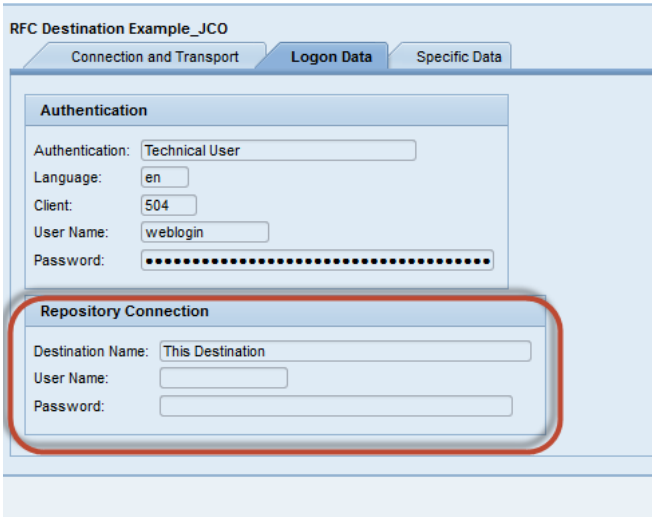
2.4.1 Integration within CRM: SSO Authentication

In case the CRM backend system provides an **SSO cookie** - containing a valid ticket for logon to the issuing system - this ticket is used for session authentication. In that case, the logon ticket will be assigned to a prior defined destination at runtime. However, this **destination must be already valid** at design-time, which means it

must be able to **connect to the repository** of the ABAP backend system at least. Accessing the repository allows to retrieve basic metadata information.

Destinations with valid repository access can be configured in tab *Logon Data*, by:

- Pointing to a different destination which already has repository access
- OR Entering a valid user that has sufficient authorizations



Destination Configuration - Repository Connection

- OR Maintaining a technical user used for authentication with appropriate authorizations

> **Note**
>
> Note 460089 documents the minimum authorizations required for accessing an ABAP repository.

Direct Connections to a Specific Application Server

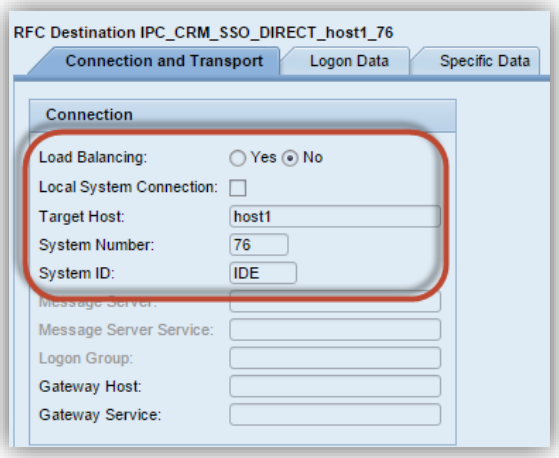
The integration into CRM (this applies to the integration into the product simulation as well as into CRM Order) requires the Product Configuration UI to connect to the same application server instance.

The configuration instance is prepared in the Configuration Engine in the CRM and lives in a session context (in a Virtual Machine Container session). Instantiating a configuration is an expensive step. To avoid multiple redundant instantiations, a data exchange service offered by the Virtual Machine Container is used. This data exchange server requires two sessions, which exchange data, to live on the same application server instance. Therefore, the Configuration UI must establish a (Stateful) connection to the same application server where the leading CRM Order session is running.

> **Note**
>
> This is standard behavior, and has been covered in earlier versions with configuration *server_connect* in XCM. The specific server instance (host and system number) will be dynamically overwritten during runtime.

As of WEC 7.54, ONE destination for EACH application server has to be configured in advance!

Destinations to a certain server can be configured in tab *Connection and Transport:*



Destination Configuration: Direct Connection Mode

Summary

For application crm/ipc integrated into CRM, the destinations should provide: **authentication with SSO to specific application server instances.** By default, **pooled connections** are used.

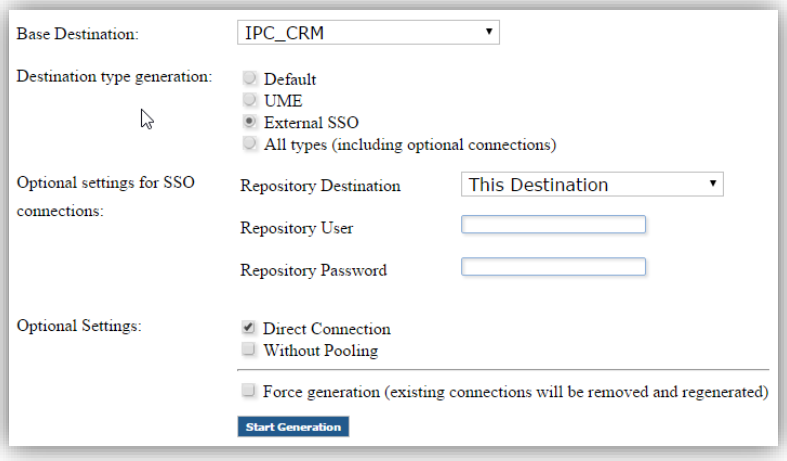
Following the naming conventions introduced in section 2.2, the destination names shall look like this:
<BaseDestination>**_SSO_DIRECT_<host>_<sysNo>**

Recommendation: Generate Destinations

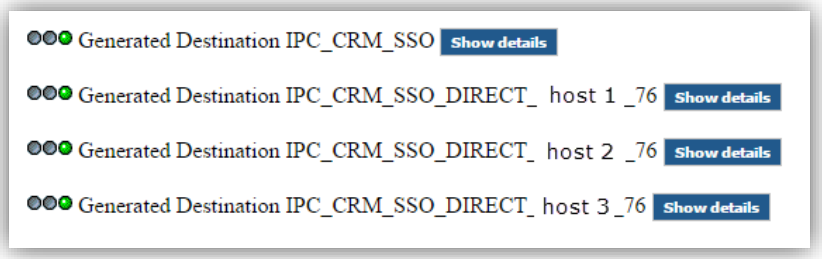
To provide destinations for all application server instances in advance, it is recommended to **generate the destinations**, as described in section 2.3.

The following options are required:

1. *External SSO* as Destination Type
2. *Direct Connection* as Optional Setting



This leads to the following list of generated destinations:



Destination Generation: Generated Destinations with SSO Authentication & Direct Connection Mode

If a new application server instance is added to the CRM system, the respective destination for this server instance needs to be added to the list of destinations. This destination can be re-generated or provided manually.

2.4.2 Integration within CRM: Authentication with Technical User

If the CRM does not provide a SSO ticket, a default connection is used.

Caution

If the **SSO logon ticket is not available**, the destination must contain **technical user** information for authentication. Repository access is not sufficient in that case.

If the **SSO ticket is not valid**, the **authentication fails** in every case!

Following the naming conventions introduced in section 2.2, the destination names shall look like this:
<BaseDestination>**_DIRECT_<host>_<sysNo>**

For the automated generation of destinations, select:

1. *Default* for Destination Type
2. *Direct Connection* as optional setting

2.4.3 Integration within ERP: Authentication with Technical User

For using the crm/ipc application integrated into the ERP, a destination with authentication type *Technical User* is sufficient, if the user has required authorizations to call the Configuration and Pricing Engine in the ERP (see note 412309). There is no need to create additional destinations in this scenario if the default destination is connectable and pooled connections are sufficient.

3 Migration Steps from JCo 2.x to JCo 3.0

This chapter targets users who have created customer applications based on versions lower than 7.54 and have implemented own RFC function calls.

WEC specific coding changes that are required to support JCO 3.0 are described here. Additionally, tips & tricks on a fast migration are illustrated.

> **Recommendation**
>
> Before performing the steps described in the following, you should make yourself familiar with the differences between JCO 2.x and 3.0 from a technical point of view.
>
> The following JCO documents contain the most important information:
>
> SAP JCo 3.0 Documentation (English) / SAP JCo 3.0 Dokumentation (German)
>
> SAP JCo Migration Guide (English) / SAP JCo Migration Guide (German)
>
> Both documents can be found at https://service.sap.com/connectors → SAP Java Connector → Tools & Services

3.1 Package and Naming Changes

JCO 3.0 is delivered within a new package. Classes (e.g. `JCO.Function`) have been refactored. The following steps are required to adapt the coding to the new structure:

3.1.1 Change DC dependencies

	Old	New
Software Component	ENGINEAPI	ENGFACADE
Development Component	com.sap.mw.jco	tc/bl/jco/api
Package Name	com.sap.mw.jco*	com.sap.conn.jco*
Class Names	JCO.*	JCo*

Structural Changes between JCO 2.x and 3.0

Recommendation

You can use **regular expressions** to replace the old class names within your coding.

RegEx Search String	RegEx Replace String
(JCO\.)(Function|Structure|Field|Table|ParameterList|Exception|Record|Attributes|FieldIterator|MetaData)	JCo\2
(JCO)(\.TYPE_)	JCoMetaData\2
JCO\.(AbapException)	\1

Regular Expressions to replace old class names starting with "JCO."

Depending on your development environment, switching to the new package imports can be performed automatically afterwards.

In case of NWDS: Right-click on the project → *Source* → *Organize Imports*

3.1.2 Resolve Build Errors

close() method no longer supported for classes of type JCoConnection

Old Syntax

```
connection.close();
```

Solution

Calling the `close()` method for a `JCoConnection` is no longer required. The connection lifecycle is handled by NetWeaver now.

destroy() method no longer supported for classes of type JCoConnection

Old Syntax

```
connection.destroy();
```

Solution

Calling the `destroy()` method for a `JCoConnection` is no longer supported.

If needed, call the `cleanup()` method for connections of type `JCoStateful` instead!

Adapt metadata related calls on JCO records and parameter lists

The metadata model for Function Parameter Lists or JCo Records has been changed. Metadata information are encapsulated in separate objects for *records* and *parameterlists*.

Old Syntax

```
custGetDetail.getExportParameterList().hasField()
```

or

```
input.getName()
```

Solution

Retrieve the Metadata Object for the specific JCo Object:

```
custGetDetail.getExportParameterList().getMetaData().hasField()
```

or

```
input. getMetaData().getName()
```

Exception Handling

The basic `JCoException` does no longer inherit from `RuntimeException`. As a result, they can only be catched in cases they were thrown explicitly before.

Issue 1

```
catch(JCoException ex)
```

might result in build errors like: *"Unreachable catch block for JCoException"*

Solution 1

It is sufficient to change `JCoException` into `JCoRuntimeException`, in case of:

```
catch (JCoException ex) {
JCoHelper.splitException(ex);}
```

Reason: Method `splitException()` is analyzing the Exception in more detail.

Case-by-case decisions are necessary.

Issue 2

```
catch(AbapException ex) {
```

Solution 2

Exceptions of type `AbapException` are explicitly thrown by function modules. Class `BackendFMException` has been introduced, which is thrown within `JCoHelper.splitException(ex);`

In case explicit handling of those function module exceptions is required, use:

```
catch (BackendFMException ex) {
```

Adapting .setValue() calls to new signature

Signature for JCo Parameterlists and Records (JCoStructure or JCoTable) has been changed.

Old Signature

```
.setValue(<value>,<fieldname>)
```

New Signature

```
.setValue(<fieldname>,<value>)
```

A plugin has been developed which identifies candidates of JCo objects for which the parameters within .setValue signature needs to be changed. For those candidates the switch of parameters can be performed with the JCo Migration Helper.

3.2 JCo Migration Helper

The JCo Migration Helper is an NWDS/Eclipse plugin. It is designed to help customers to migrate from JCo 2.x to JCo 3. This plugin is open source and can be downloaded and installed from https://github.com/SAP/jcomigrationhelperplugin. The documentation and the feature list is available there as well.

4 Troubleshooting

This chapter targets users who run into connection errors while using WEC 7.54.

Continue reading this chapter, in case you find an entry similar to the following within your log file:

com.sap.isa.core.eai.JCoDestinationException: Creating a connection for destination "XY" failed.

```
at com.sap.isa.core.eai.sp.jco.JCoManagedConnectionFactory.retrieveDestination(…)
```

OR

com.sap.isa.core.eai.JCoDestinationException: Creating a connection for connection definition with key "XY" failed. Respective destination could not be determined.

```
at com.sap.isa.core.eai.sp.jco.JCoManagedConnectionFactory.retrieveDestination(…)
```

> **Note**
>
> See Viewing Log and Trace Files if you do not know how to view the log file.

The `JCoDestinationException` is the main exception thrown by the application whenever there is an issue with internal destination handling which prevents the application from communicating with the backend system. However, this is just the wrapping Exception.

> **Important**
>
> It is important to find the causing exception of JCoDestinationException in order to identify the actual root cause!

Causing Exceptions could be:

- `DestinationNameDeterminatorException`
- `DestinationPropertiesMissingException`
- `BackendRepositoryException`
- `JCoException` especially of type: `JCO_ERROR_RESOURCE, JCO_ERROR_LOGON_FAILURE, JCO_ERROR_COMMUNICATION`

Those exceptions will be explained on the following pages including suggestions on how to solve them:

1) Caused by: com.sap.isa.core.eai.sp.jco.DestinationNameDeterminatorException: The extended destination name could not be determined. Reason: Base destination name is empty! You may check the configuration in Extended Configuration Management (XCM)

```
at com.sap.isa.core.eai.sp.jco.JCoManagedConnectionFactory.determineExtendedDestName-
FromProperties()
```

Description:

This is thrown, when the base destination name is empty. Without that name it is not possible to determine the full destination name for which an communication instance can be retrieved.

Reason:

Destination Name is not maintained in XCM

Solution:

Check that the right base destination value is maintained for parameter destination.default. It is specified within the jco component of your XCM scenario configuration!

Please refer to chapter 2 for further configuration details.

2) Caused by: com.sap.isa.core.eai.sp.jco.DestinationPropertiesMissingException: User/Password Details not available. Establishing a Stateful JCo Connection to the Backend System is not possible!!!

```
at com.sap.isa.core.eai.sp.jco.JCoManagedConnectionFactory.checkUserLoginIsRequested()
```

Description:

This is thrown, when the application relies on user and password information for setting up a Stateful Connection after a successful application login, which is not available in the connection properties.

Reason:

User and Password information got lost or has never been transferred correctly after application login.

Solution:

In case you have extended the connection handling by overwriting configuration files, like backendobject-config.xml or eai-config.xml, you need to check your changes. Backend objects that are defined to use Stateful connections rely on user/password information for login type ISA_LOGIN.

If you did not change connection configuration and the issue is reproducible, please open a message for component CRM-ISA-TEC with the stacktrace and reproduction example attached.

3) Caused by: com.sap.isa.core.eai.BackendRepositoryException: Error occured while retrieving JCoRepository from destination: XY

```
at com.sap.isa.core.eai.sp.jco.JCoManagedConnectionFactory.retrieveRepositoryFor-
Destination()
```

Description:

This is thrown, when the according JCoRepository for a destination is not accessible or could not be created.

Possible Reasons:

- No user/password is maintained as Technical User or as Repository User
- In case a user is set: The maintained User does not have sufficient authorization
- In case a different reference destination is set: The destination is not valid for accessing the repository

Solution:

Check your destination settings within destination service and user authorizations according to details in chapter 2.4.1

4) Caused by: com.sap.conn.jco.JCoException: (106) JCO_ERROR_RESOURCE: Destination XY does not exist

```
at com.sap.isa.core.eai.sp.jco.JCoManagedConnectionFactory.checkIfDestinationExists()
```

Description:

The destination service is not able to return an existing destination instance for the given name.

Reason:

This usually occurs if the base destination exists but for the extended destination name (derived from connection properties during runtime, e.g. authentication type) a destination has not been created yet.

Solution:

Check the list of existing destinations within the destination service and make sure that a destination with the name mentioned in the stacktrace is available and connectable.

Recommendation: Use the generation tool (introduced in chapter 2.3) to create additional destinations in addition to the base destination (it can be assumed that the base destination entered in XCM is valid)

5) Caused by: com.sap.conn.jco.JCoException: (103) JCO_ERROR_LOGON_FAILURE: Initialization of destination XY failed: The system is unable to interpret the SSO ticket received on SID mshost host1

Description:

This is thrown when one of the SSO login types (UME_LOGIN or SSO_LOGIN_ABAP) is active and the retrieved destination cannot be initialized for communication due to incorrect authentication.

Reason:

The ticket string - usually provided by the NW system (in case of UME or LDAP authentication) or by the ABAP system (in case of external SSO login) - is not accepted by the destination service. It is not

Solution:

This issue is usually related to misconfiguration of one the systems that provides the SSO. Depending on the ticket issuing system you may check Configuring the AS Java to Accept Logon Tickets related documentation that covers UME/SSO authentication mechanisms.

6) Caused by: com.sap.conn.jco.JCoException: (102) JCO_ERROR_COMMUNICATION: Initialization of destination XY failed: <Reason, e.g. Connect to message server host failed>

Description:

This is thrown when the retrieved destination instance is not able to connect to the respective backend system.

Reason:

Communication errors can have multiple reasons. Those are further described within the error message, e.g. *Connect to message server host failed*

Solution:

Check the detailed error message. It usually points to incorrect destination properties which have to be corrected within the destination service.

www.sap.com/contactsap