Loftware ABAP Package V4.5

Label Printing in SAP using Loftware Automation Implementation Guide

Rev-7

Contents

Introduction

An SAP transport package, formerly known as "development class", groups multiple related development objects in a single distributable unit that enables transferring data from one SAP installation to another. Whenever you create a new development object (such as a table, program, etc.) that you want to publish on another SAP server, you must assign it to a package. After being assigned to an object, the object is placed into a transport request.

The ABAP Package contains objects that provide a framework for communicating with Loftware Automation back-end label-printing service. ABAP Package exposes the API that is utilized by SAP applications to send the data for printing. Once the data has been received from the SAP application, the Package encapsulates it in an XML message and sends it to Loftware Automation for processing. You can use Web Service (SOA) or HTTP (RFC type G) communication to Loftware Automation.

Besides label printing, the ABAP Package can also provide label preview (PDF, PNG, or JPEG), report the live status of the label printer, generate a list of all labels in the document management system (DMS), provide a list of available printers, a list of variables defined in the label and provide a binary print job into SAP.
NOTE: the printer must support reporting its live statuses.

This document describes the necessary steps to import the ABAP Package to your SAP system and to configure it.

Distribution of the ABAP Package

You receive the ABAP Package as a ZIP file with the following content:

- Folder **ABAP Transport Package**. This folder contains the transport package itself. For details on how to transport the package, see chapter **Importing ABAP Package transport request** on page 16.
- Folder **Automation configuration**. This folder contains the Automation configuration (MISX file) and sample labels. We provide a generic sample label and some sample labels that contain all data sources from two standard SAP transactions (VL02N and CO02).
    - **`Label.nlbl`** is a generic sample label used throughout this document.
    - **`DeliveryNote.nlbl`** is a label for outbound deliveries, when you want to print each item from a delivery note on a separate label.
    - **`DeliveryNoteReport.nlbl`** is a label for outbound deliveries, when you want to print all items from a delivery note in table/report as a shipping document (e.g., A4 or Letter page size). The .CSV and .VALUES files are sample data files for previewing this label in Loftware Designer.
    - **`ProductionOrder.nlbl`** is a label for production orders.
- Folder **Documentation**. This folder contains the technical documentation, such as this document.

Requirements

The following requirements apply to Loftware:

- Loftware Cloud Business or above for cloud or NiceLabel LMS Enterprise for on-premise.
- Loftware software installed in the default location.[1]
- PDF reader installed on Windows computer with SAP GUI (to see PDF label previews).

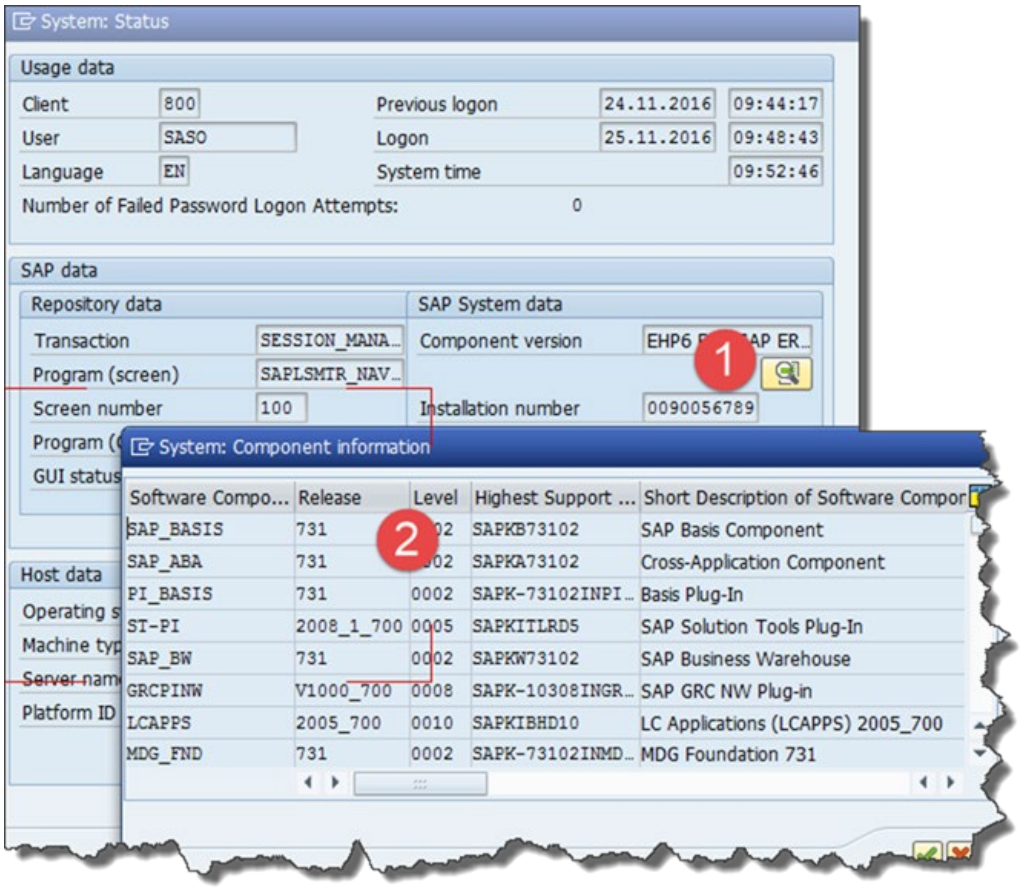
The following minimum requirements apply to SAP:

1. SAP products based on the NetWeaver stack (e.g., ECC/ERP, SCM, EWM).
   Enhancement Package 6 (EHP6)
   or
   On-premise SAP S/4HANA
   or
   SAP S/4HANA Private Cloud

2. Unicode SAP system

To verify the component version, do the following:

1. In SAP GUI, select **System>Status**.
2. In **System:Status**, click the button **Component information**.
3. **SAP_BASIS** and **SAP_ABA** must be at least in release **731**.



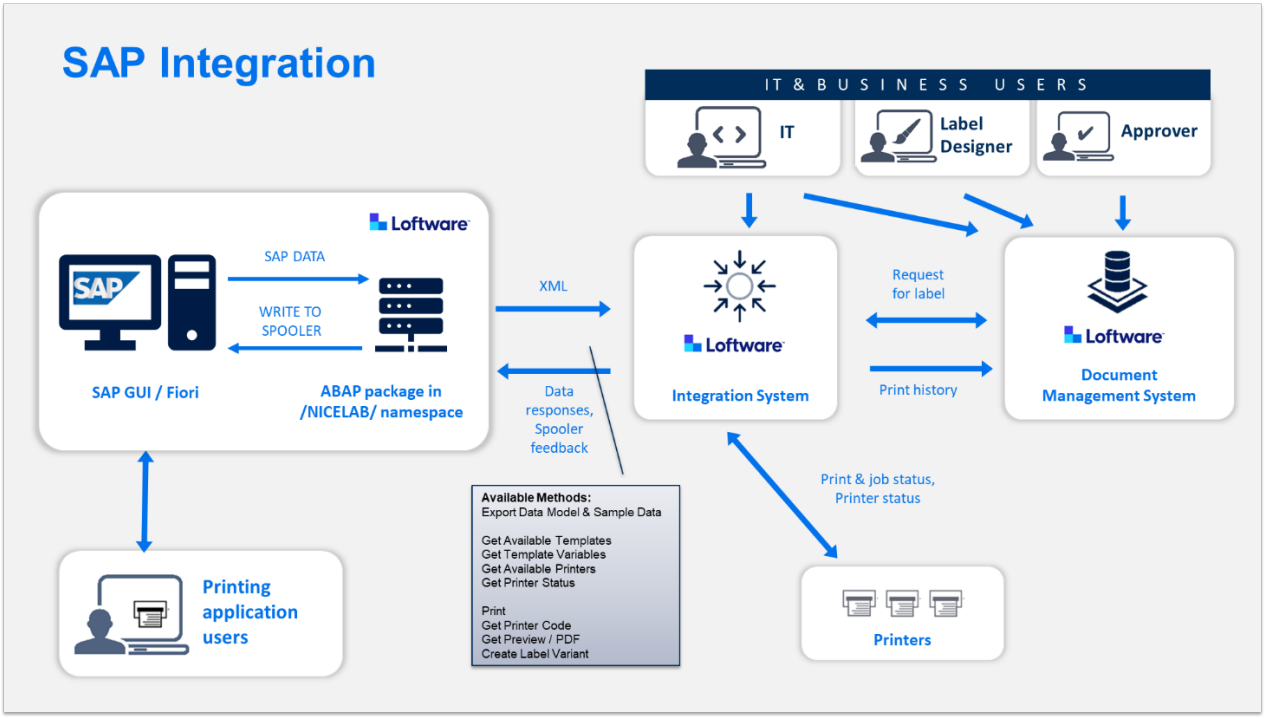
[1] If you use non-default location, you must update the Automation configuration with your non-default path.

Architecture

Overview

ABAP Package is a collection of below-listed software components that simplify integration of label printing from SAP:

1. ABAP transport package that you install in SAP.
2. Pre-configured Automation Enterprise with SOA (Web Service) and RFC type G (HTTP) triggers. You can configure the ABAP Package to transfer the data to Automation Enterprise using one of the two communication types. Additionally, there is a trigger to create the label catalog (a list of all labels in your DMS).
3. Sample labels prepared with data sources for printing from some standard SAP transactions (such as Outbound Deliveries - VL02N).



ABAP Package is transported into the SAP system and is available within the /NICELAB/ namespace.

ABAP Package is the connector that binds together the SAP and Loftware ecosystems. On one hand, ABAP Package receives commands and data from SAP applications, while on the other hand, it communicates with the Loftware integration system – Loftware Automation.

For better understanding, you can look at ABAP Package as an API for label printing. ABAP Package works like an SDK (Software Development Kit) for label printing from SAP. It exposes certain methods that you call from your SAP applications.

The request received from the SAP application is encapsulated in the XML message. ABAP Package sends the XML content to the integration system (Loftware Automation) for processing. Depending on what the user wants, Loftware Automation runs various actions. For example, the user might request a label print action, generation of a label preview, generation of the binary print job, label catalog, information about the selected label, list of available printers, or label printer's live status.

When printing labels, Loftware Automation loads the specified label from DMS, merges it with values for data fields as received in XML, and uses the Loftware printer driver to prepare a print job in Windows Spooler. For example, if you use Zebra printers, the result is a ZPL (Zebra Programming Language) file in the Windows Spooler.

NOTE: By default, SAP Spooler is not involved in the process in any way. It is the responsibility of Windows Spooler on the computer with installed Loftware integration system to send data to the printer. Optionally, you can also command Loftware Automation to provide the binary print job back to SAP Spooler, so you can also send it to the printer from SAP.

The computer with installed Loftware Automation must meet the following prerequisites:

1. **The drivers for your label printers must be installed.** You can install printer drivers directly on this computer, or you can register the drivers from the print server you might have.
2. **All printers must be visible to the computer and to the user account under which Automation Service runs.** Loftware Automation on this computer creates print jobs and sends them to the printers using Windows Spooler. The printers must be accessible for printing.

You create the label templates in Loftware Designer, and you store them in the DMS, where they are governed by the approval workflows and version control system. DMS also supports role-based access control (RBAC) and stores the history of all print events (e.g., all name-value pairs are remembered together with the revision number of the printed label, printer name, and label copies). You can also reprint labels from the DMS.

Loftware Automation configurations

The ABAP Package bundle provides pre-built Loftware Automation connectors (triggers). You do not have to configure anything on the Automation's side. You must just deploy the provided configuration and start the triggers within.

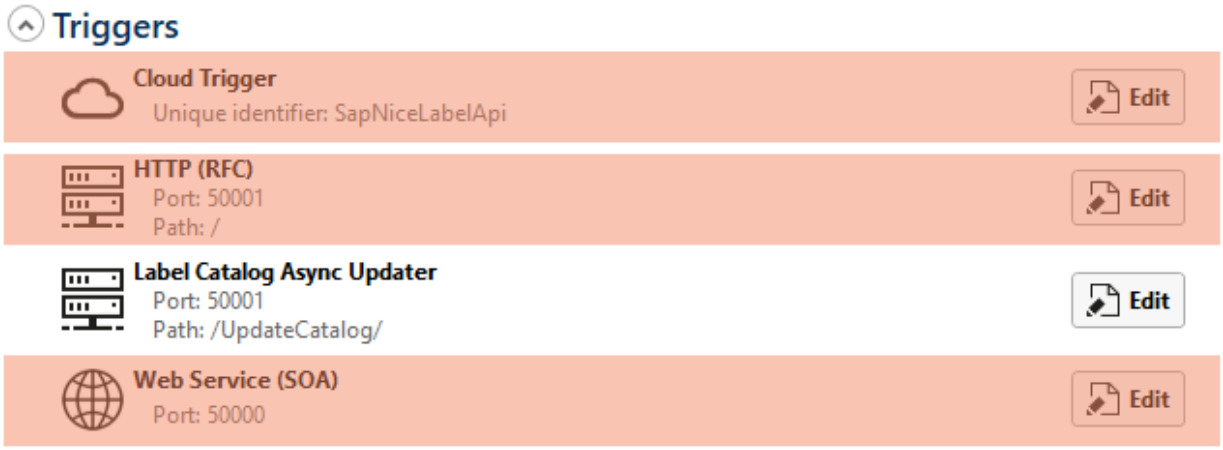
The ZIP bundle contains two Automation configurations.

- **SAP Loftware API V4.5 LC.misx.** Use it with your Loftware Cloud account. The configuration also contains the Cloud trigger (for cloud-to-cloud connectivity).
- **SAP Loftware API V4.5 NLLMS.misx.** Use it with your Loftware LMS Enterprise product.

Each Loftware Automation configuration contains a few triggers that all provide the same functionality. The difference between triggers is in the communication type between SAP and Loftware Automation. Choose the communication type that fits your environment best.

1. **SOA (Web Service).** Running on port 50000/TCP. It accepts Web Service requests from the ABAP Package.
2. **RFC type G (HTTP).** Running on port 50001/TCP. It accepts HTTP requests from the ABAP Package.
3. **Cloud trigger (technically, this is also RFC type G (HTTP).** In this case, SAP does not communicate directly with the Automation server (there might not be a direct connection possible). SAP sends the message to Cloud Trigger API that acts as a proxy and forwards the data to your Automation server running behind a firewall. This trigger allows cloud-to-cloud connectivity. You do not have to open any ports on the computer with Automation or the company's firewall.
4. **Label Catalog Async Updates.** This trigger is available in the configuration variant for NiceLabel LMS Enterprise only. This trigger help to generate a list of labels from the Document Management System.

- The configuration variant for Loftware Cloud uses internal Document API to generate the catalog and does not need this trigger.

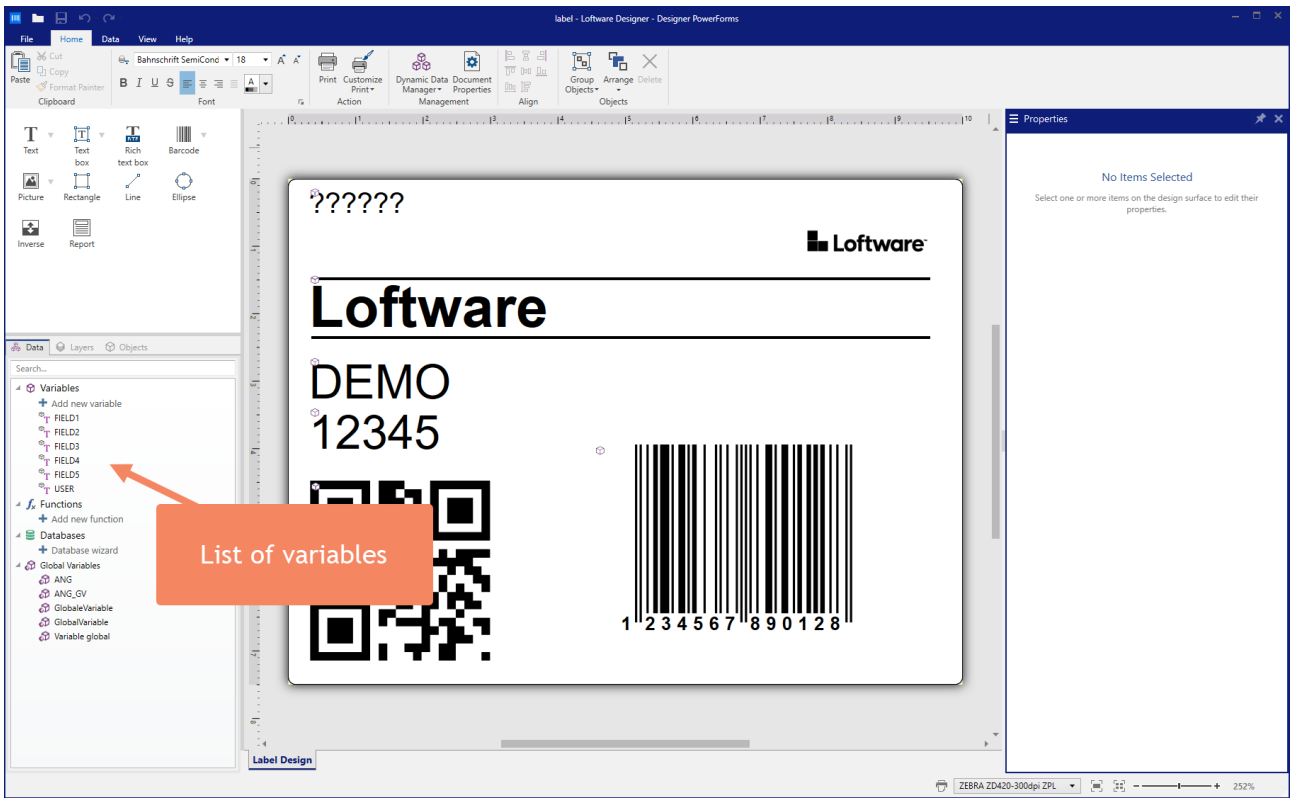


NOTE: If necessary, you can change the port numbers on which the Automation triggers respond. Open the configuration in Automation Builder and edit it.

Label design

Use Loftware Designer to create and maintain labels. Loftware Designer runs in a Microsoft Windows environment. This is a graphical designer with an intuitive user interface that business users themselves can use to create dynamic labels. When you create a dynamic label, you configure label objects to receive data from data sources called “variables”. When ABAP Package sends the SAP data for printing into Loftware Automation, that data will be saved in the variables and printed on a label.

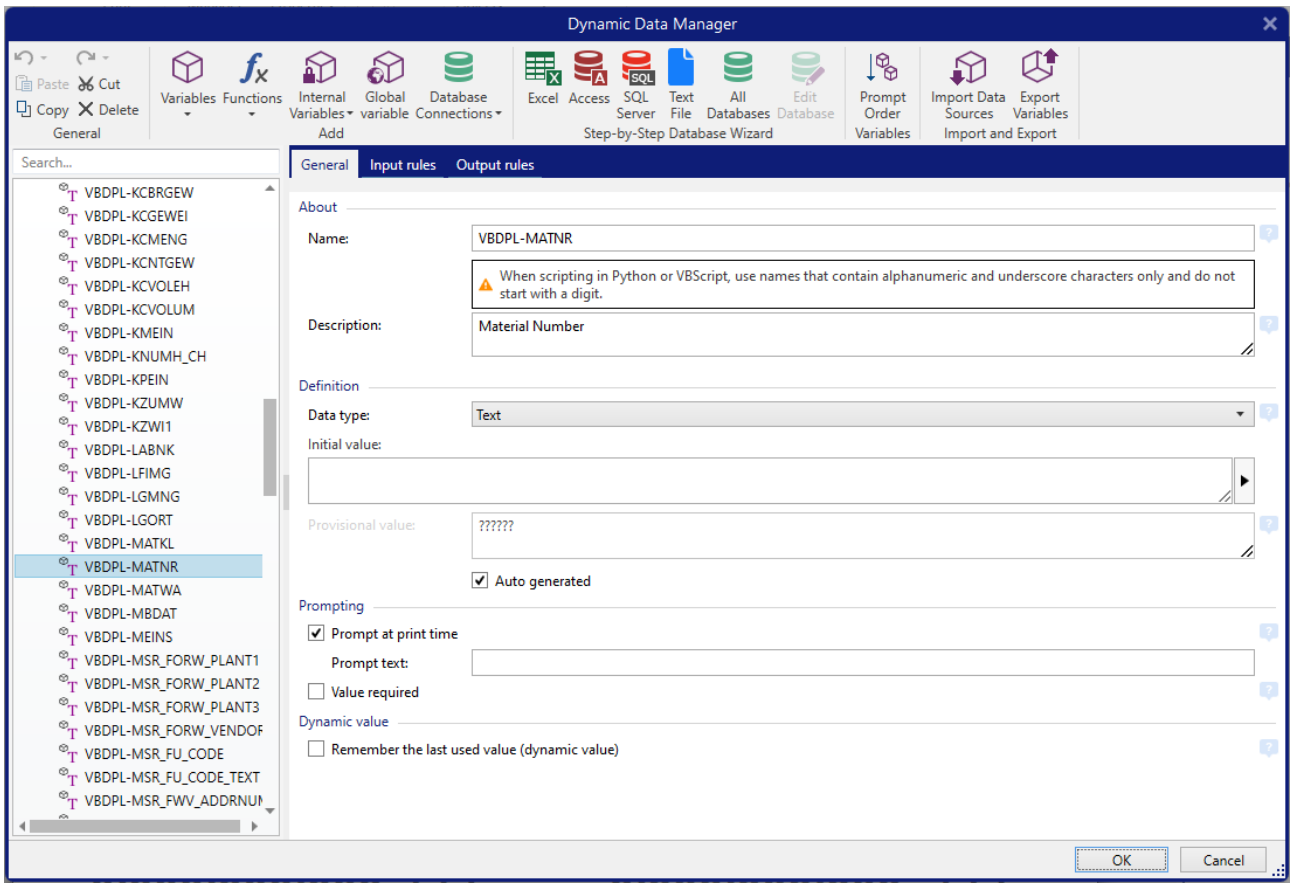
When you open a sample label template **`label.nlbl`** (provided with the ABAP Package) in Loftware Designer, you can see the label contains variables, such as FIELD1, FIELD2, FIELD3, etc. These variables are linked with some of the label objects and provide values for those objects at print time.



Variables must have the same names as the data fields in the SAP application from which you execute label printing. You can create variables on the label manually, you can import variables from existing labels, or you can import variables from other supported data files (such as XML).

The ABAP Package contains sample labels for some SAP standard transactions (such as VL02N). The sample labels already contain **all variables** that exist in the associated transaction. You can immediately use the variables as data sources for your label objects. You can also import these pre-defined data sources into other labels.

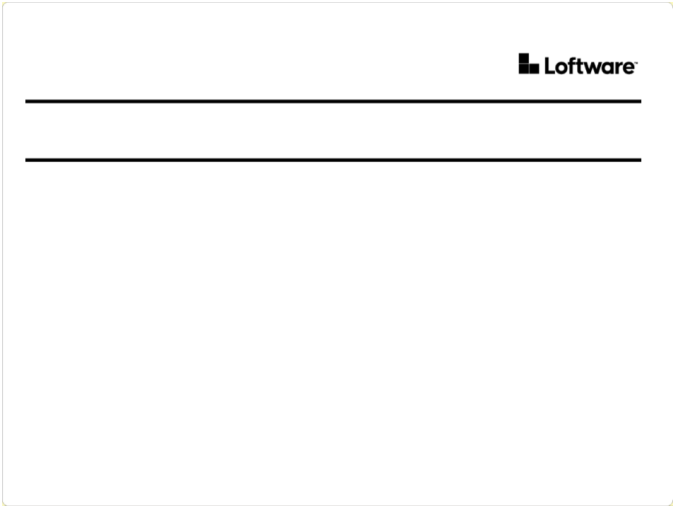
For example, the **`DeliveryNote.nlbl`** label contains all variables from the outbound deliveries transaction (VL02N). There is a prefix “VBDPL” defined for each variable, as used in SAP.



Label print and data mapping

If you do not provide any value to label variables, the sample label template **`label.nlbl`** prints like the following.

1. All dynamic label objects that expect to receive some data inputs are empty as no value has been provided for them.
2. Only static label objects get printed.



To print data on a label, you must send data (name-value pairs) from SAP into the ABAP Package.

When ABAP Package receives data (name-value pairs) from the SAP application, it encapsulates them in XML and sends them into Loftware Automation for processing. The name-value pairs are saved in the “Data” element in the XML message. Loftware Automation extracts all name-value pairs and sends them to the label.

NOTE: For more information about the XML structure and data exchange between ABAP Package and Loftware Automation, see the chapter **Data exchange** on page 91.

Data mapping is performed by the matching names:

1. If a field in XML finds a match in the list of variables defined in the label, that variable gets populated with the value from XML.
2. If the match is not found in the label that field from XML is ignored.

For example, if you provide the following data into the demo transaction:[2]

FIELD NAME	FIELD VALUE
FIELD1	Loftware
FIELD2	DEMO
FIELD3	12345
FIELD4	1234567890
FIELD5	123456789012

The label printout looks like this.

Provided values have been assigned to the label variables on the same name and then used with the label objects.

[2] Learn more about how to use demo transaction in the chapter Configuring the ABAP Package.

Setting up the integration

Deploying Loftware Automation

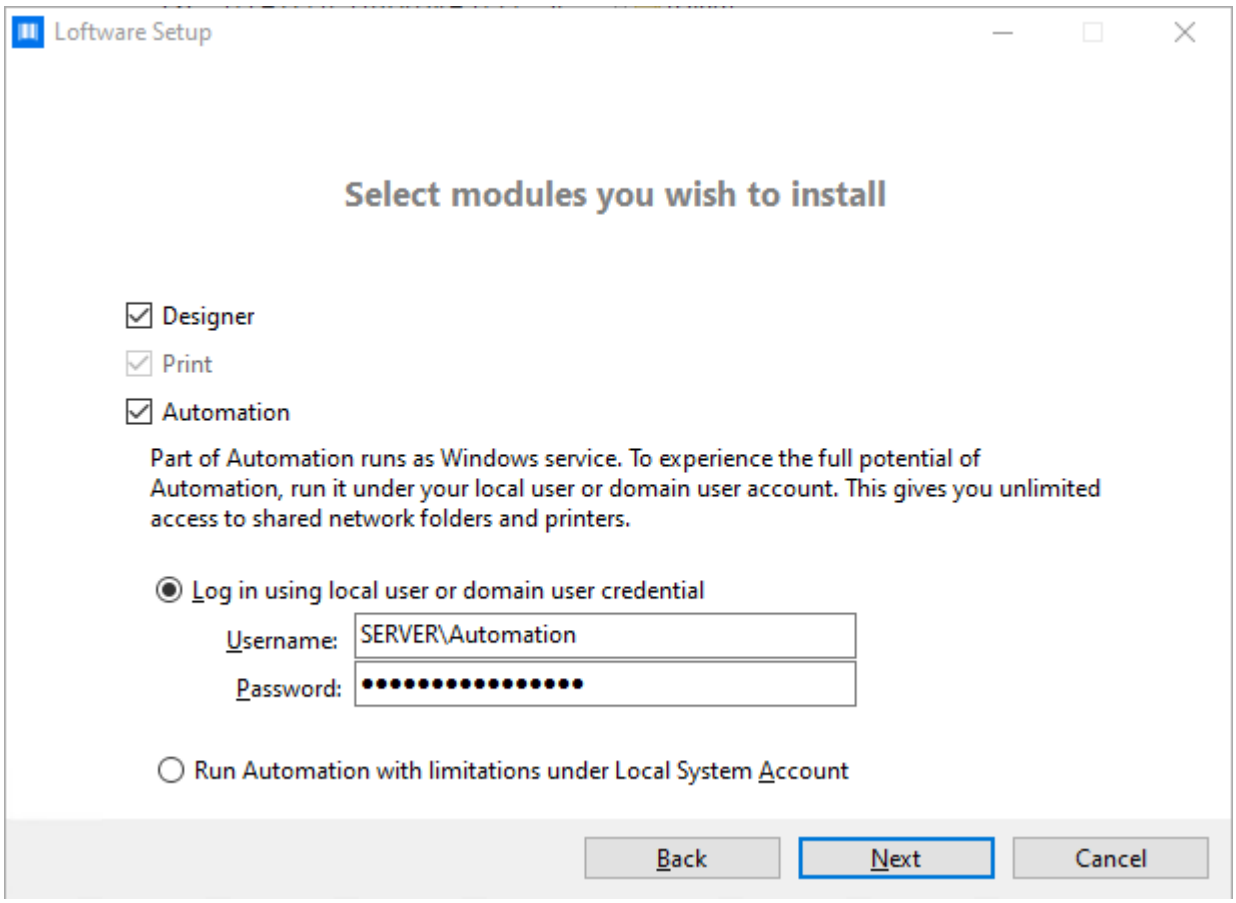
The ABAP Package ships with pre-configured Loftware Automation. To deploy the configuration, you must load the configuration into Loftware Automation Manager, start the triggers and reconfigure Windows Firewall to allow inbound connections on ports associated with the configuration.

Loftware Automation comes pre-build with two different configurations. Make sure to load and activate the one that matches your software product.

Installing Loftware Automation

Loftware Automation is bundled in the installation of Loftware desktop applications. The required module to run the Automation configuration for the ABAP Package is “Automation”. It will run as the Windows service application.

NOTE: The best practice approach is to run the Automation Service under credentials of a real user account, not Local System account.



For detailed Loftware installation steps, see Loftware Installation Guide.

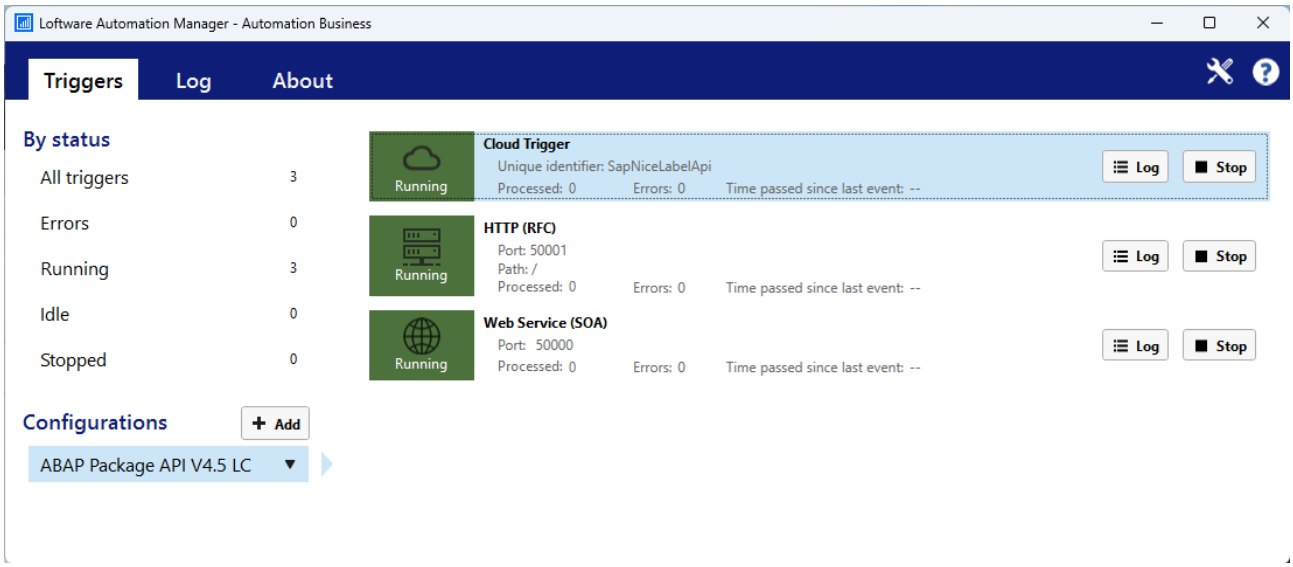
Deploying triggers

Do the following:

1. Go to the computer, where you have installed Loftware Automation.

2. Make sure you have activated the Loftware software.
3. Save all Loftware Automation files to a folder. You must copy all files from the **Automation Configuration** folder from ZIP file. Use the Automation folder that matches your Loftware product (Loftware Cloud or NiceLabel LMS).
4. Run **Loftware Automation Manager** application.
5. In the **Triggers** tab, click the **+Add** button.
6. In the **Open** window, browse to your **`.MISX`** file and click **Open**.
   Automation Manager loads the configuration and lists the triggers in the right-hand pane.
7. Select all triggers and click **Start** on any of the triggers. All triggers will start. Triggers' icons turn into a green color and the status change to "Running".
   a. You can later deactivate the triggers you don't need. Typically, you would enable just one trigger type.

NOTE: Loftware software must be activated or running in a trial mode to be able to start the triggers.



Configuring Windows Firewall

You must configure the firewall on a Windows computer to allow inbound communication to ports configured for SOA (Web Service trigger) and RFC type G (HTTP trigger). You must allow connection to the port for the communication mode you intend to use. Typically, you would open just one port.

NOTE: You do not have to open any port if you plan to run the Cloud trigger.

To create an inbound rule, do the following:

1. On a computer running Automation, on the **Start** menu, choose **Control Panel**, choose **System and Security**, and then choose **Windows Firewall**.
2. In the navigation pane, choose **Advanced settings**.
3. In the **Windows Firewall with Advanced Security** window, in the navigation pane, choose **Inbound Rules**, and then in the Actions pane, choose **New Rule**.
4. On the **Rule Type** page, choose **Port**, and then choose the **Next** button.
5. On the **Protocol and Ports** page, choose **Specific local ports**, and then enter the port number you want to open.
   a. If you plan to use SOA (Web Service trigger) communication, open port 50000.
   b. If you plan to use RFC type G (HTTP trigger) communication, open port 50001.

6. Click **Next**.
7. On the **Actions** page, select **Allow the connection** and click **Next**.
8. On the **Profile** page, choose the profiles, and click **Next**.
9. On the **Name** page, type a name for the rule, and click **Finish**.

Configuring TLS/SSL encrypted communication (HTTPS traffic)

Loftware Automation configuration that ships with the ABAP Package does not have SSL/TLS communication enabled. The data will be exchanged over the regular HTTP traffic.

If necessary, SSL/TLS can be enabled by reconfiguring the provided Automation configuration. For more information about how to configure Automation, see the chapter **Using Secure Transport Layer (HTTPS)**.

Generating a subscription key to consume Loftware Cloud APIs

NOTE: Cloud-to-cloud connectivity is not available for on-premise NiceLabel products.

You need a subscription key to consume APIs available with Loftware Cloud Business (or above) products. A subscription key is used for API authentication in the Loftware software. Loftware Cloud runs on multi-tenant software architecture. The Loftware Cloud API endpoint created with the Automation configuration for ABAP Package is the same for all Loftware customers. When you activate the Cloud Trigger in the provided Automation configuration, it will create an endpoint https://labelcloudapi.onnicelabel.com/Trigger/v1/CloudTrigger/SapNiceLabelApi in your Loftware subscription. When some other customer activates the same Cloud Trigger, the same endpoint will be created for them. Based on the subscription in your HTTP request the API tells the customers apart and forwards the message to the correct Automation server.

You need a subscription key in the following cases:

- Enable cloud-to-cloud connectivity between your SAP system and Loftware software.
- Use PI/PO integration and your Automation is not deployed in the same LAN as your SAP system, so you need to use Cloud trigger API to access Automation's endpoint.
- Use CPI integration and want to access Automation’s endpoint that might be deployed behind firewalls or disparate networks.
- Get a list of available label templates (Label Catalog).
- Get a list of available cloud-connected printers.
- Print labels to cloud-connected printers.

Generate a subscription key

This is a summary of the steps you must complete to get a subscription key:

1. Add a **Cloud Integration** in your Control Center (**Integrations>Cloud Integrations**).
2. Create an account in the **Developer Portal**.
3. Register the **Cloud integration** from Control Center in the **Developer Portal**.
4. In Developer Portal, create a product **Loftware Cloud**.
5. Activate the product **Loftware Cloud** to receive a subscription key.

The detailed guide to complete the above steps are provided in the chapter Cloud integrations in the Control Center user guide.

Add items (custom IDs) to the configuration table

The next step is to add two custom items to the configuration table.

1. Run transaction **/NICELAB/IF_CTRL**.
2. Add item **API-VERSION** with value “v1”.
3. Add item **OCP-APIM-SUBSCRIPTION-KEY** with the value of the primary or secondary key you received from the Developer Portal (for the Label Cloud subscription you created).

For details see **Configuring system-wide defaults** chapter in this document.

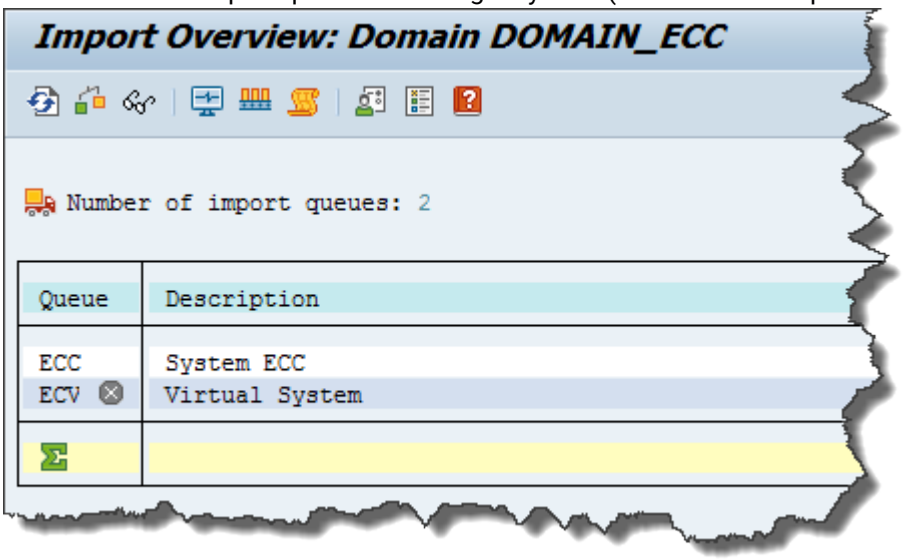
When the above two Custom IDs exist in the configuration table, the ABAP Package will always include them in the request to Loftware Automation in two places:

- **In the custom HTTP headers of the HTTP request.** These are authentication requirements so that the ABAP Package can deliver data into Cloud trigger published in Loftware Automation.
- **Inside the <Header /> element in the XML payload.** These are authentication requirements for Automation to use Loftware Cloud APIs (to generate the label catalog, to get a list of cloud-connected printers, and to print labels to cloud-connected printers).

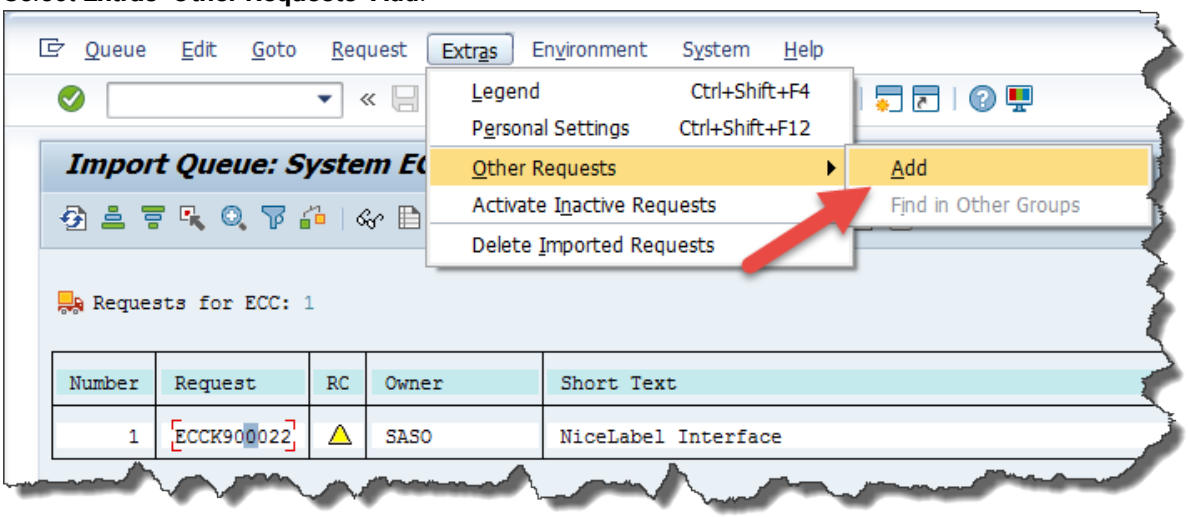
Importing ABAP Package transport request

To import the ABAP Package transport request, do the following:

1. Copy the transport package files into the appropriate folder on your SAP system.
   a. The R*.ECC file goes into `/usr/sap/trans/data` folder.
   b. The K*.ECC file goes into `/usr/sap/trans/cofiles` folder.
2. In **SAP GUI**, run the transaction **STMS**.
3. Click **Import Overview**  .
4. Double-click the import queue of the target system (**ECC** in the example below).

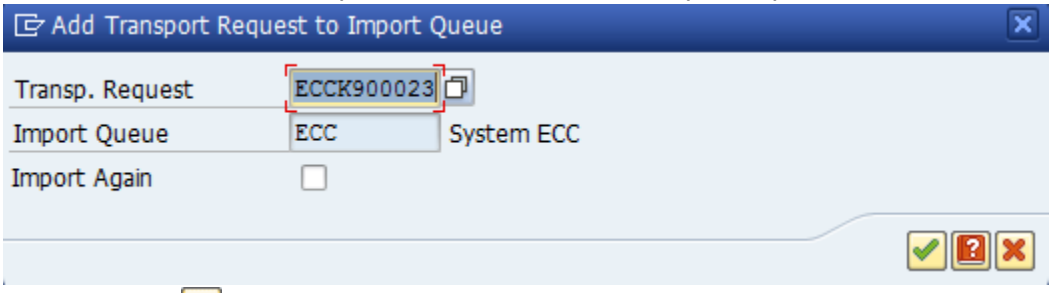


5. Select **Extras>Other Requests>Add**.



6. Enter the **Transp. Request** number.

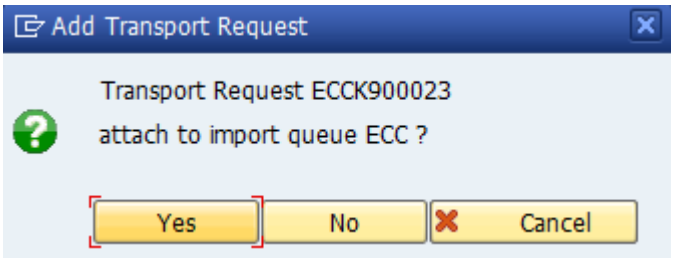
   You can click the Search Help and search for the transport request in the list.



7. Click **Continue**.

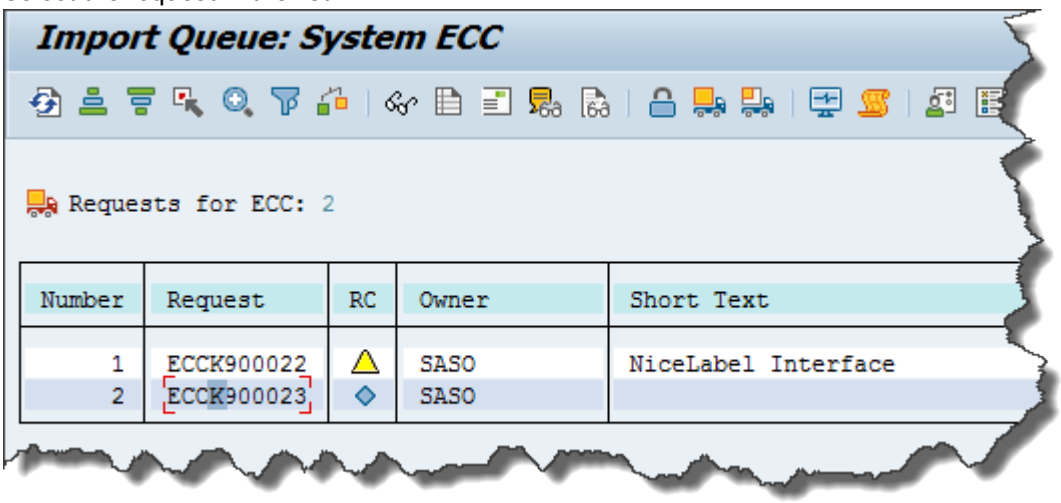
   The **Add Transport Request** window opens.

8. Click **Yes**.

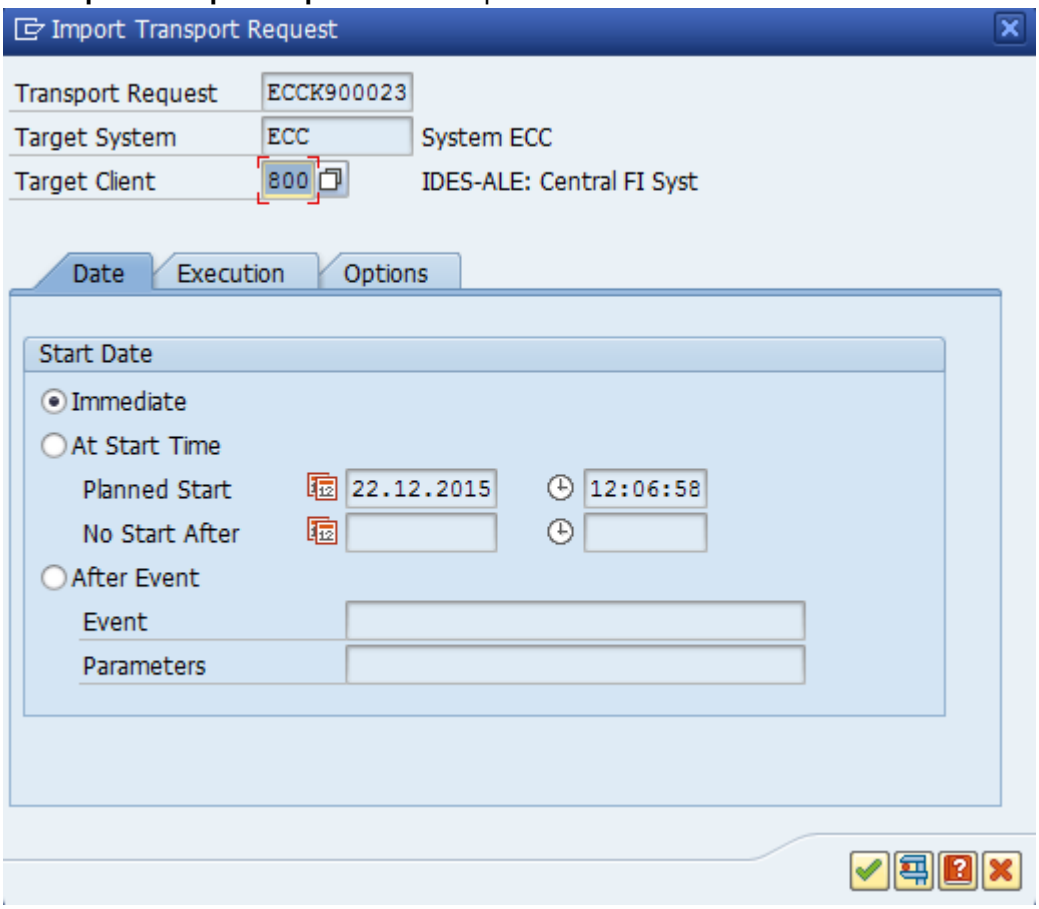


   The request is added to the import queue.

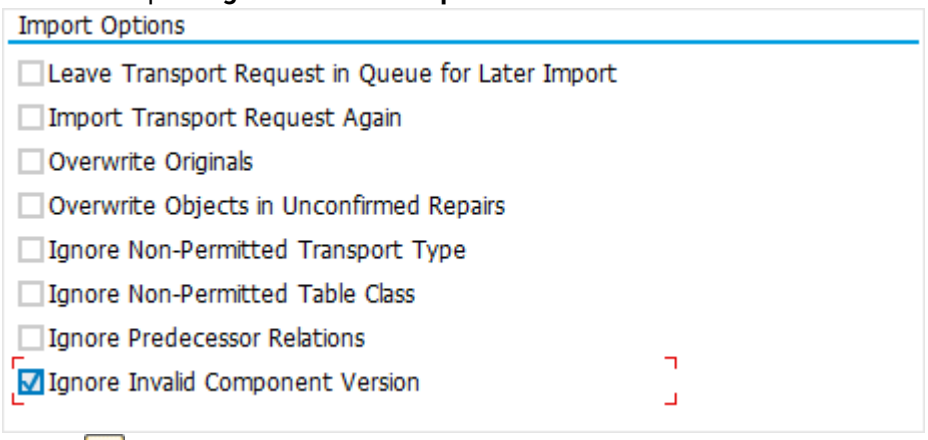
9. Select the request in the list.



10. Click **Import Request** .
    The **Import Transport Request** window opens.

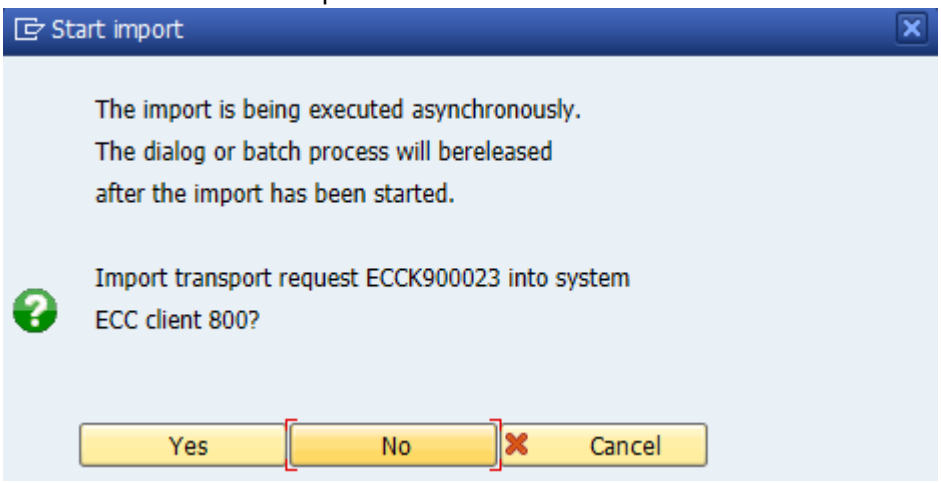


11. In **Target Client**, enter the client number.
12. If you are importing the transport request on the **SAP S/4HANA** system, also do the following:
    a. Go to the **Options** tab.
    b. Enable the option **Ignore Invalid Component Version**.

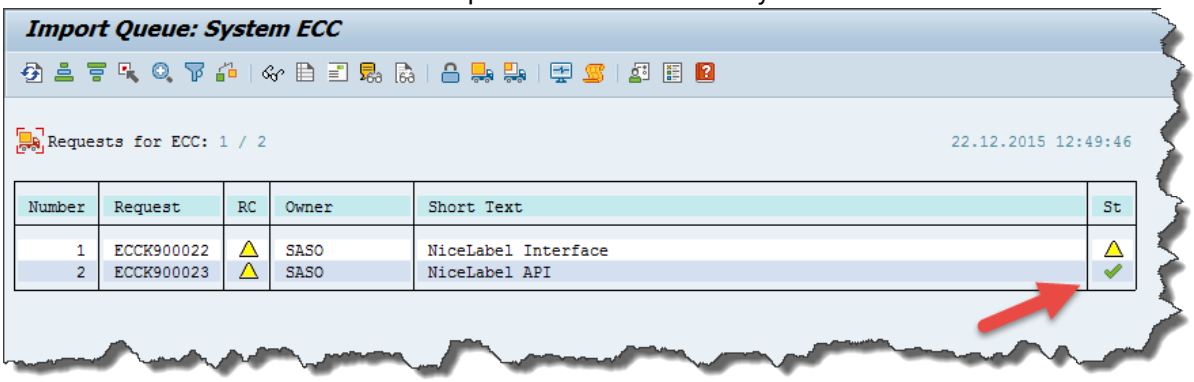


13. Click **Continue** .

14. Click **Yes** to confirm the import.



15. The checkmark confirms that the import finished successfully.



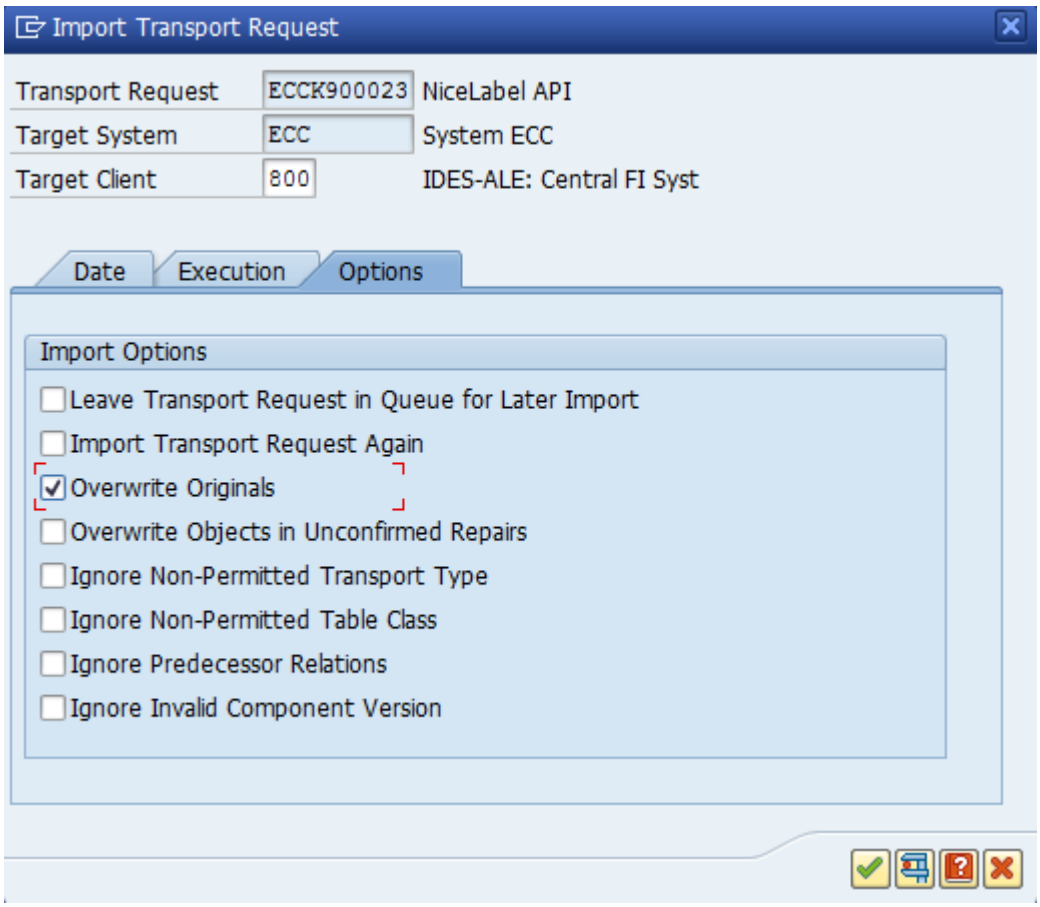
This import must be performed by an SAP Basis administrator.

Upgrading the ABAP Package

When upgrading the ABAP Package to a newer version, follow the instructions for the transport request import as provided in the topic Importing ABAP Package transport request on page 16, but with a single exception.

In the **Import Transport Request** window, go to the **Options** tab and make sure the option **Overwrite Originals** is selected.

NOTE: If you are importing the transport request on the **SAP S/4HANA** system, also enable the option **Ignore Invalid Component Version**.



Incompatibilities between versions

V2

- **Location of the constants.** In V1, the constants were saved in the /NICELAB/CL_INTERFACE_ROOT class. In V2, constants were moved to public interface /NICELAB/CL_IF_UTIL. If you reference the constants in your custom classes where you use the ABAP Package, you will have to update the custom code to point to /NICELAB/CL_IF_UTIL.

V3

- **Recreating logical ports for Web Service consumer proxy.** In V3, ABAP Package uses a new consumer proxy. You have to recreate the logical port when you use Web Service communication.

V4

- **Support for multi-server landscape.** ABAP Package V4 supports multiple Automation endpoints, not just one. The process configuration (business rules) table (**/NICELAB/V_IF_PR**, also accessible via tcode **/NICELAB/IF_PROC**) is expanded to support the endpoint configuration. Each item in the table also defined the field OperMode, where you define the type of the endpoint, and the field Endpoint ID, where you define the name of the endpoint. For example, the OperMode **HTTP Request** and Endpoint ID **EMEA** would send a request to Automation using outbound RFC call to an RFC destination EMEA (as defined in tcode SM59).

  When you upgrade to ABAP Package V4 your existing process configuration table will be expanded with new fields. The values for the new fields will be taken from your existing interface configuration (tcode **/NICELAB/IF_CTRL**), so there should be no upgrade issues.

The print programs provided with the ABAP Package have also been updated to support the new process configuration table (/NICELAB/OUTPUT_PROCESSING and /NICELAB/PSFC_OBJECT_LIST).

- **Label catalog generation.** When you have a Loftware Cloud product, Automation uses Document API to create a label catalog, not the command-line utility as in previous versions. The Document API provides a faster and synchronous response. To use the Document API, you have to provide the subscription key in your payload. Automation will use if for authentication when accessing the Document API. For more information, see the chapter **Requesting label catalog for Loftware Cloud**.
- **The content of Revisions fields in the label catalog.** In V4, Automation will return the version number of the last revision of the label template. The field will contain a single value, no longer all available versions. The returned version number depends on the access role to which the user running the Automation server belongs.

Configuring Destination in SAP

The ABAP Package collects the label printing data, encapsulates it in an XML message, and sends a request to Loftware Automation. The outbound request can use SOA (Web Service) or RFC type G (HTTP) communication type.

Configuring SOA (Web Service) destination

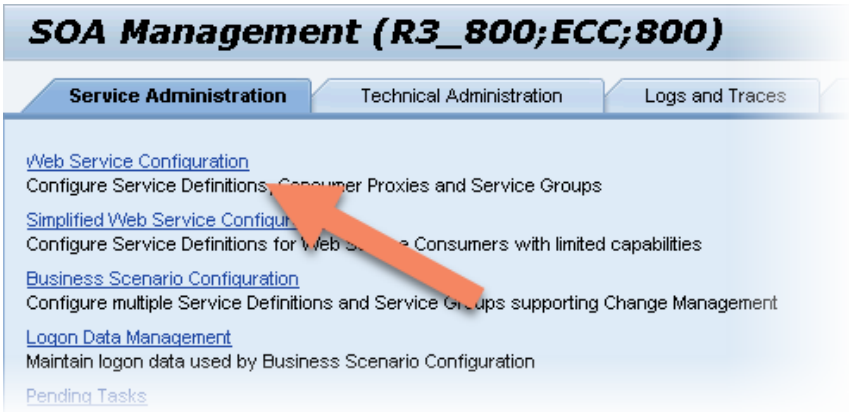
The necessary configuration step involves defining the **Logical Port**, through which the ABAP Package sends the request to Loftware Automation.

The logical port defines the transport parameters for reaching Loftware Automation. The two most important parameters are the computer name and port number. If you want to use a Web Service destination, you must define at least one logical port.

Configuring the logical port

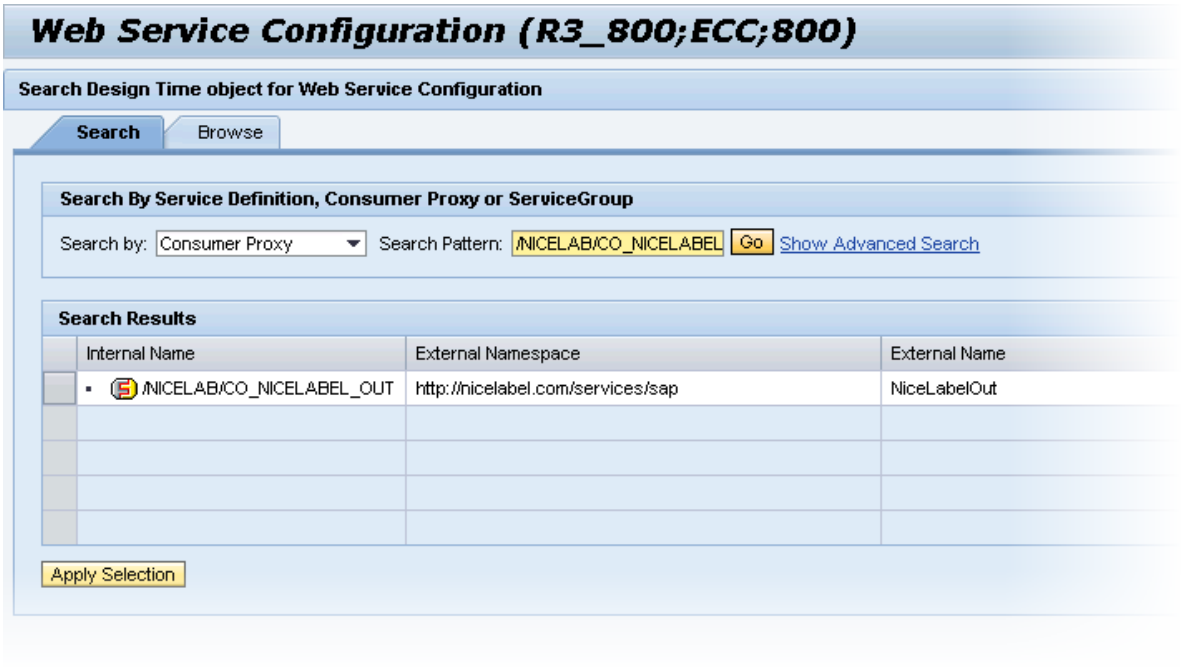
To configure the logical port, do the following:

1. In **SAP GUI**, run the transaction **SOAMANAGER**.
   The **SOA Management** opens in the browser.
2. In the **Service Administration** tab, click **Web Service Configuration**.

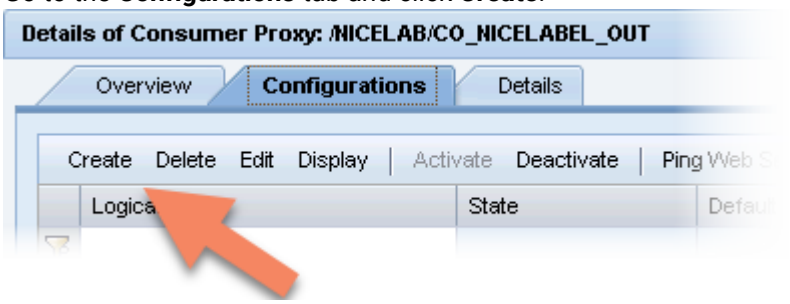


3. In **Search by** drop-down box, select **Consumer Proxy**.

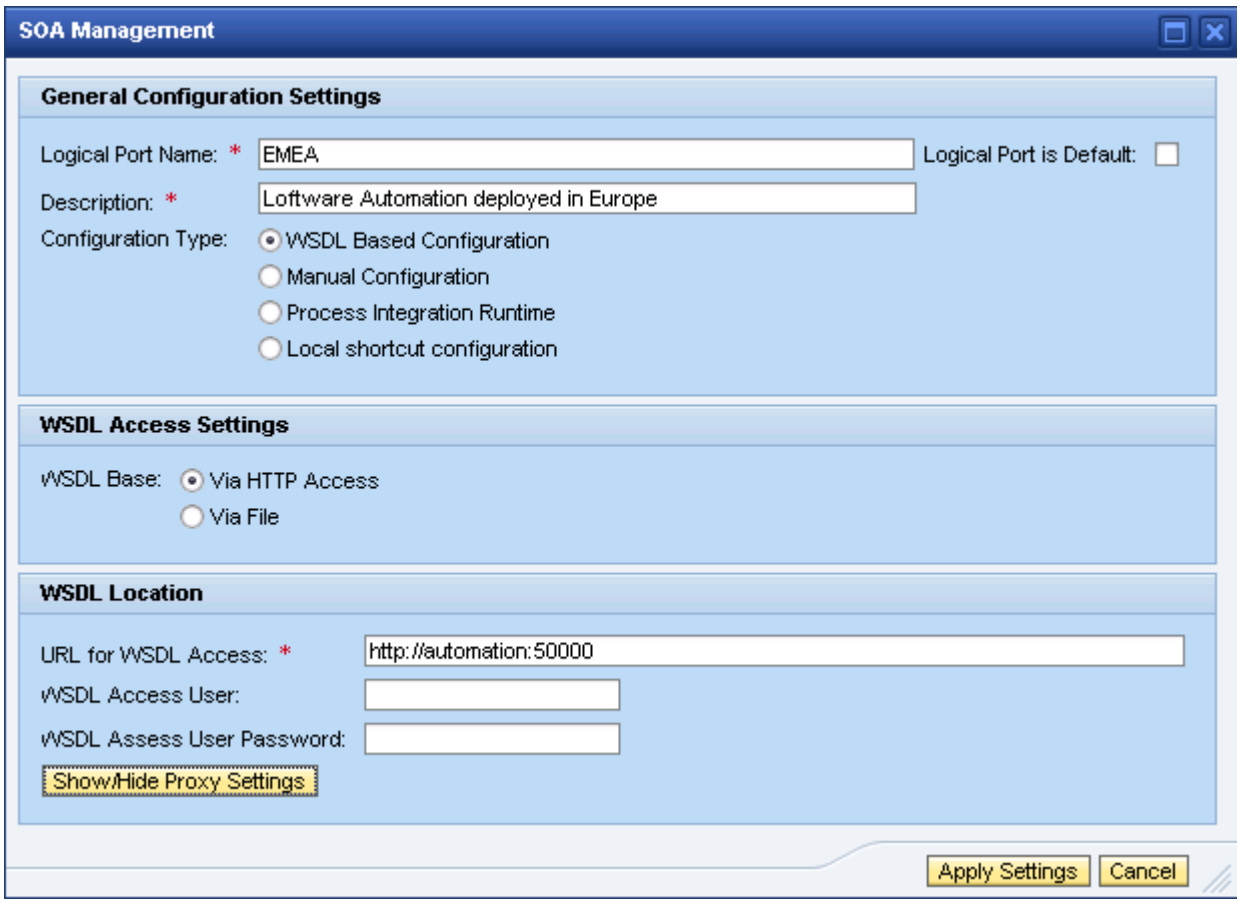
4. In **Search Pattern,** enter the name of the consumer proxy as provided by the ABAP Package. Enter **/NICELAB/CO_NICELABEL_OUT** and click **Go**.



5. Select the **/NICELAB/CO_NICELABEL_OUT** entry in the search results and click **Apply Selection**. The consumer proxy details are displayed.
6. Go to the **Configurations** tab and click **Create**.



A new SOA Management window opens displaying the configuration details.
7. In the **Logical Port Name** field, enter the name that identifies this logical port.
8. In the **Description** field, enter the text that describes the logical port.
9. For **Configuration Type**, select **WSDL Based Configuration**.
10. If you have access to the Loftware Automation and if the Web Service trigger has already been deployed, select **Via HTTP Access** for **WSDL Base**. In **URL for WSDL Access**, enter the path to the Web Service trigger as published in the Loftware Automation.



In this example, Loftware Automation is installed on the Windows server with name **automation.** The Web Service trigger has been configured on port **50000**.

11. You can also import the WSDL from a file. Initially, you must use a web browser to open the URL to Loftware Automation, load the WSDL from it and save the content to a file.

    To get WSDL and save it to disk, do the following:

    a. Open the following URL in a browser:
    http://<automation>:50000/?wsdl

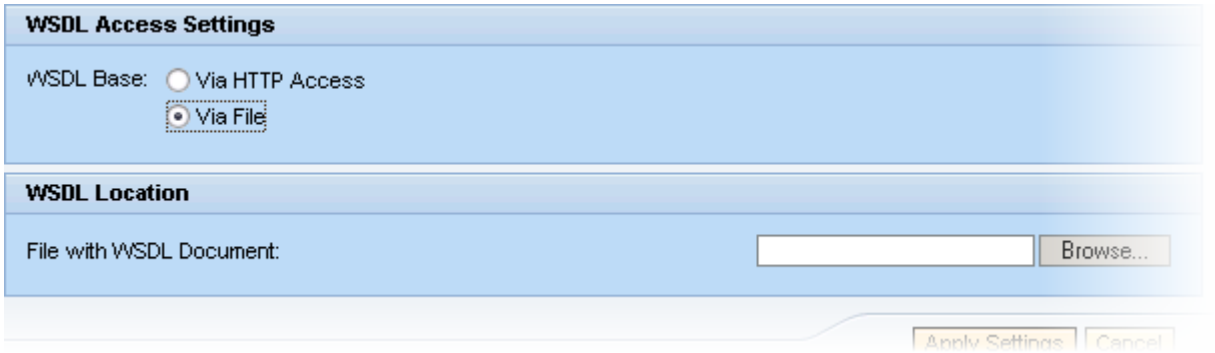
    Replace the name `<automation>` with the name of the Windows computer, where you have installed Loftware Automation.

    b. Right-click the content in the browser and select Save As. Save the WSDL content to file on local disk.

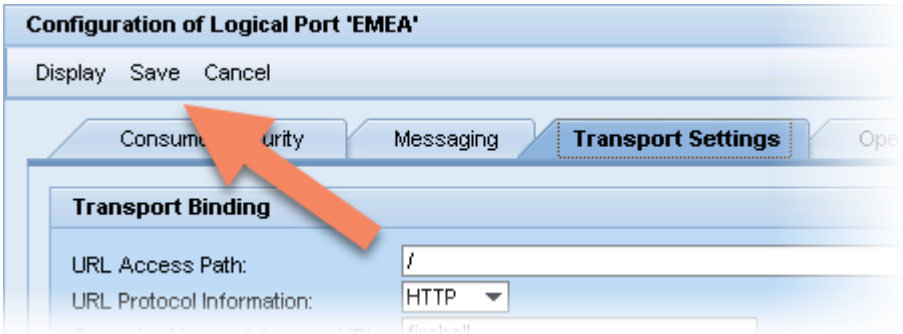
    In SOA Management, select **Via file**, then click **Browse** and select the WSDL file you have just saved.

    IMPORTANT: No matter which option you use to get the WSDL (Via HTTP Access or Via File), the URL endpoint in the WDSL will contain the hostname of the server where Automation is installed. This is fine, when you connect to the Automation using local area network. However, if Automation is installed outside of LAN and you need fully qualified domain name (FQDN) to connect, make sure you update the URL endpoint to match your FQDN.

    NOTE: When you click the **Apply Settings** button to apply the WSDL file, you must have the communication path to Loftware Automation open, or SOAMANAGER will report a problem “Error in WSDL parsing: Exception occurred in library handler”.



12. Click **Apply Settings**.
    The additional logical port configuration appears.
13. Review the parameters. Normally, you do not need to change anything.
14. Click **Save**.



Testing the logical port

To test the configured logical port, do the following:

1. In the **Details of Consumer Proxy** section, select the logical port you have created.
2. Click **Ping Web Service**.
3. Scroll up to the top of the browser window to see the feedback from the test.
4. You should see the following message:

   Web service ping failed (RC=400). Service Ping ERROR: Bad Request

   This tells you the message was successfully sent to the Web Service in Loftware Automation, just the request was not formed in the structure as expected by Loftware Automation.
5. If you see the following message:

   SRT Framework exception: Service Ping ERROR: Error when calling SOAP Runtime functions: SRT: Processing error in Internet Communication Framework: ("ICF Error when receiving the response: ICM_HTTP_CONNECTION_FAILED")

   it tells you that the connection to the Web Service in Loftware Automation is not possible. You will have to resolve the problem.

   Verify the following problems:

   a. The triggers in Loftware Automation are not started.
   b. Communication to Loftware Automation is not possible. Verify the firewall settings on the computer with Loftware Automation to allow inbound connectivity.

      For more information, see **Configuring Windows Firewall** on page **14**.
   c. Is there a system-wide policy applied to the SAP server to prevent outbound connectivity to non-authorized destinations?

Configuring HTTP (RFC) destination (on-premise)

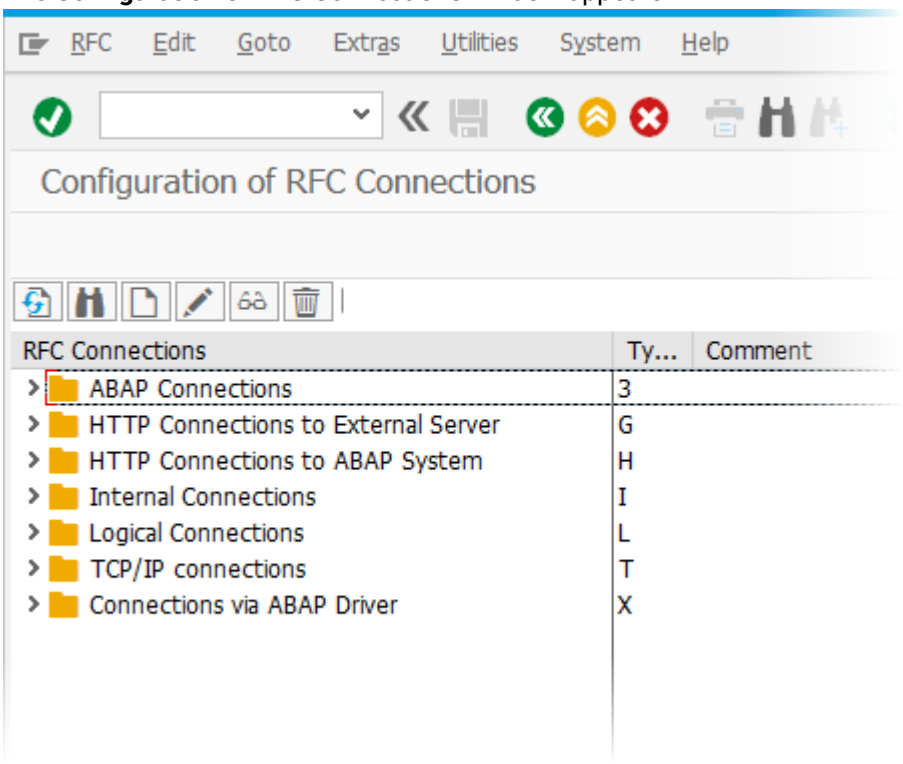
The necessary configuration step involves defining the HTTP destination to which the ABAP Package sends the request to Loftware Automation. The two most important parameters are the target host (computer name) and service number (port number).

In this scenario, SAP can communicate directly with the Loftware Automation. Both are running in the same network and connection to the Automation's IP address and configured port number is possible.

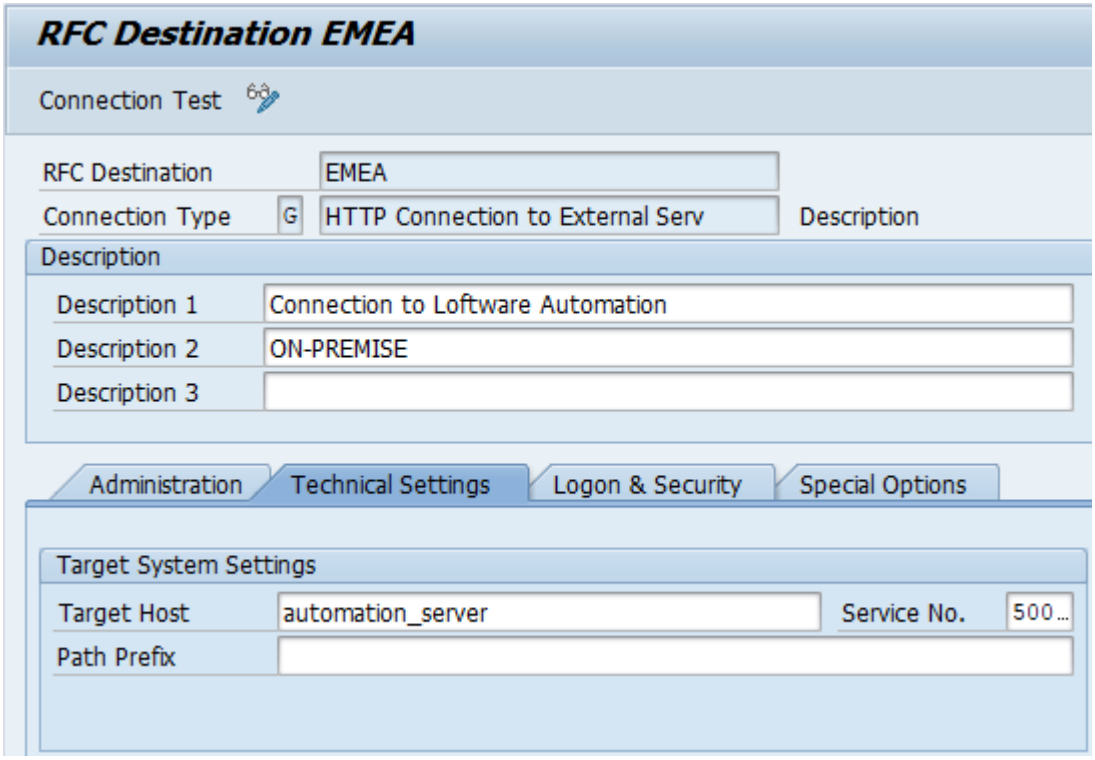
Configuring destination

To configure the HTTP (RFC) destination, do the following:

1. In **SAP GUI**, run the transaction **SM59**.
   The **Configuration of RFC Connections** window appears.



2. Click **Create**  .
3. In **RFC Destination**, enter the name of your destination. Use a descriptive name that will later be used to reference the target Loftware Automation server.



4. For **Connection Type**, select **G**. This is the HTTP Connection to External Server.
5. In the **Description** section, enter a comment to describe what this destination will be used for.
6. Go to the **Technical Settings** tab.
7. In the **Target Host**, enter the host name of the server with installed Loftware Automation.
8. In the **Service No.**, enter the port number, on which the HTTP trigger has been configured in Loftware Automation. The configuration provided with ABAP Package defines the HTTP trigger on port **50001**.
9. In **Path Prefix**, enter the path, on which the HTTP trigger has been configured in Loftware Automation. The configuration provided with ABAP Package uses a root path (/).

If you need to update the existing HTTP (RFC) destination, run the transaction **SM59** again, select your RFC destination in the **HTTP Connections to External Server** list and click **Change** .

Testing the destination

To test the RFC type G (HTTP) destination, do the following:

1. Make sure the Loftware Automation has been configured and enabled as documented in the chapter **Deploying Loftware Automation**.
2. In the **SM59** transaction, open properties of the HTTP (RFC) destination you have just configured.
3. Click the **Connection Test** button.
4. You receive the following:

   Status HTTP Response: 500
   Status text: An error occurred while trying to execute the "Use Data Filter" action. The error occurs while executing the filter "XML". Cannot load the XML file <…>"! Error in XML format. System error message: Root element is missing.

5. This is perfectly OK, even if you have not received status 200 and text OK.

   You have received a response from Automation, although an erroneous one. SAP connection test does not contain any valid data that Automation trigger can recognize; therefore Automation

responds with an error.

This response is proof that the RFC type G (HTTP) destination is configured correctly and you can send data to Loftware Automation.

Configuring HTTP (RFC) destination (Cloud trigger)

NOTE: You must have a Loftware Cloud Business subscription (or higher) to run Cloud trigger.

The necessary configuration step involves defining the HTTP destination to which the ABAP Package sends the request to Loftware Automation. The most important parameters are target host (computer name), service number (port number), and path prefix.

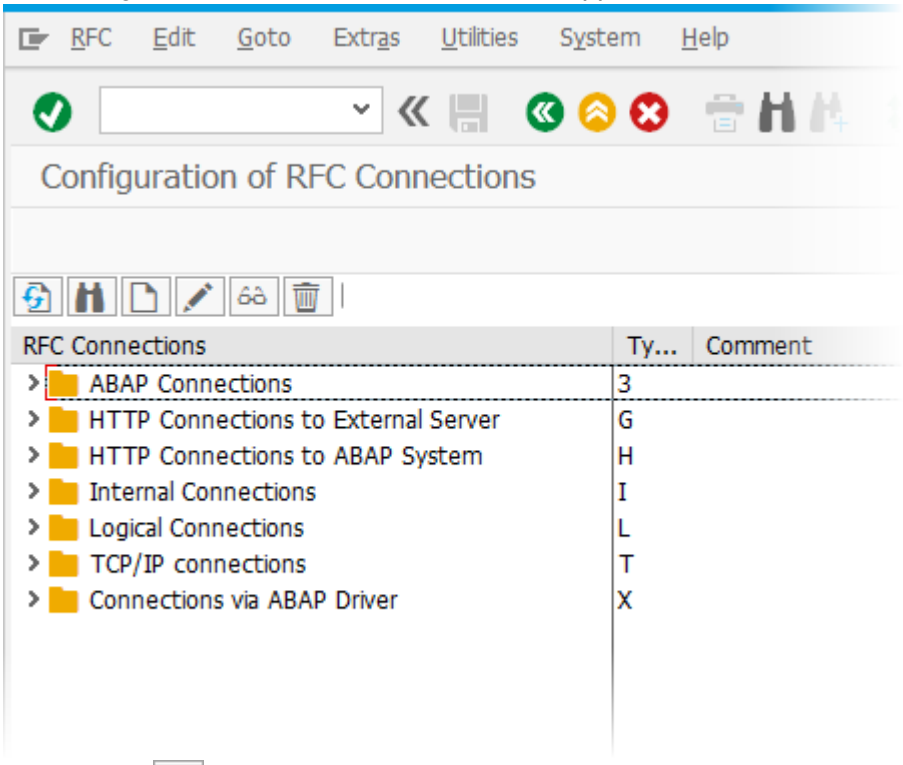
In this scenario, SAP cannot communicate directly with the Loftware Automation. A typical scenario might be SAP S/4 HANA (on-premise variant) running in the hosted virtual environment without a VPN connection down to your location. Loftware Automation is installed on-premise and has a connection to the label printers, but SAP cannot send messages to it directly.

Cloud trigger acts as a proxy in Loftware Cloud. On one hand, Loftware Automation establishes an outbound connection towards Loftware Cloud, on the other hand, SAP sends the message into the Loftware Cloud API. The message is then forwarded down to Loftware Automation for processing.

Configuring destination

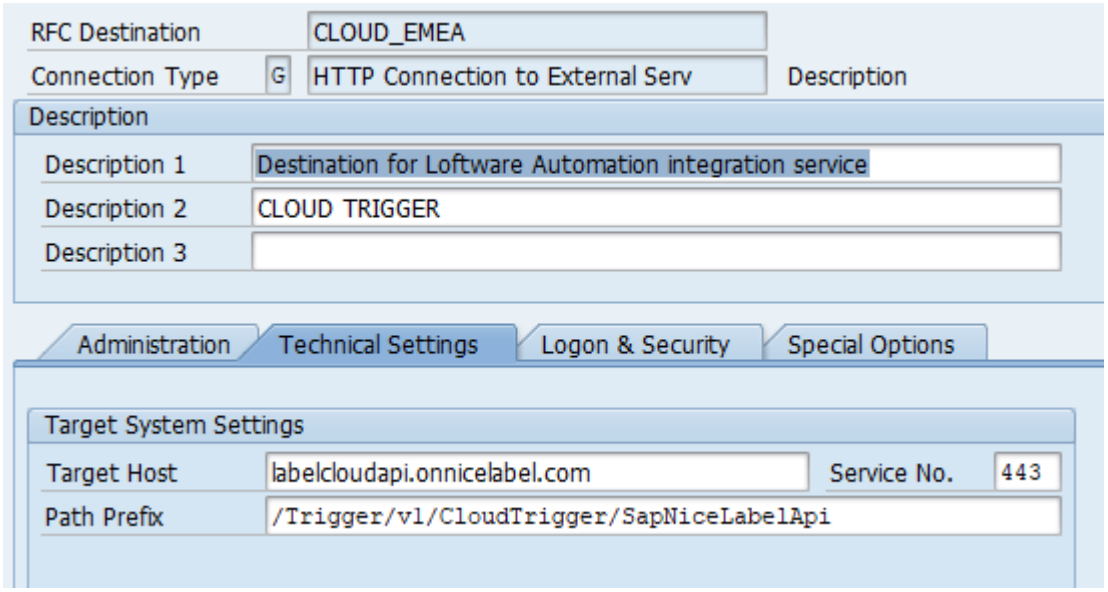
To configure the HTTP (RFC) destination for Cloud trigger, do the following:

1. In **SAP GUI**, run the transaction **SM59**.
   The **Configuration of RFC Connections** window appears.

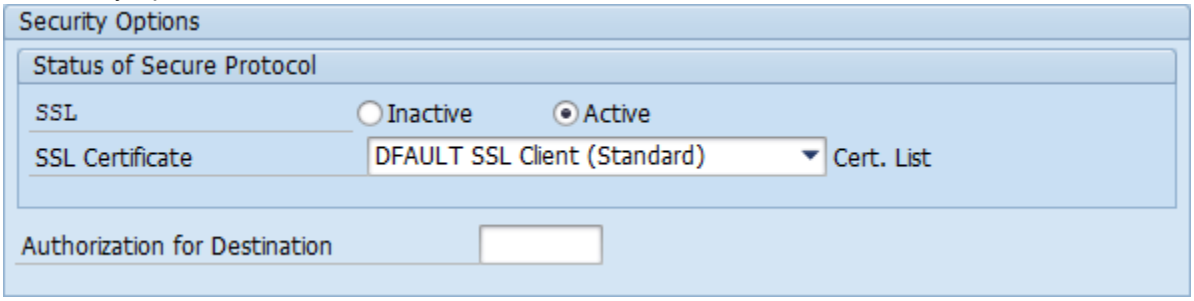


2. Click **Create**  .

3. In **RFC Destination**, enter the name of your destination. Use a descriptive name that will later be used to reference the target Loftware Automation server.



4. For **Connection Type**, select **G**. This is the HTTP Connection to External Server.
5. In the **Description** section, enter a comment to describe what this destination will be used for.
6. Go to the **Technical Settings** tab.
7. In the **Target Host**, enter "`labelcloudapi.onnicelabel.com`".
8. In the **Service No.**, enter "`443`".
9. In the **Path Prefix**, enter "`/Trigger/v1/CloudTrigger/SapNiceLabelApi`".
10. Go to **Logon & Security** tab.
11. In Security Options, select **Active** for SSL.



If you need to update the existing HTTP (RFC) destination, run the transaction **SM59** again, select your RFC destination in the **HTTP Connections to External Server** list and click **Change**  .

The next step is to configure the custom HTTP headers that must be included with each HTTP call to the Cloud trigger. The headers are configured with the following Custom IDs in the configuration table:

1. API-VERSION
2. OCP-APIM-SUBSCRIPTION-KEY

For more information on how to obtain your subscription key and configure it for the ABAP Package, see the following:

1. The chapter **Generating a subscription key to consume Loftware Cloud APIs** in this document.
2. The chapter **Configuring system-wide defaults** in this document.
3. The chapter **Cloud integrations** in the Loftware Control Center user guide.
4. The chapter **Cloud trigger** in the Loftware Automation user guide.

TLS/SSL certificates

When you have defined your endpoint with data encryption (HTTPS), there is one additional configuration step necessary in your SAP system. You must add the certificate to SAP **Trust Manager**. Use the transaction code **STRUSTSSO2**. Your SAP Basis team should be able to help you with this task.

For **Web Service (SOA)** or **HTTP (RFC)** triggers you own the certificate and already have it.

For **Cloud** trigger, the endpoint is in the Loftware Cloud. You can download the certificate in your browser.

Do the following:

1. Open your Control Center in the browser.
2. Click the **View site information** button.
3. Dependent on your browser do:
    a. In **Chrome**, select **Connection is secure > Certificate is valid**.
    b. In **Edge**, select **Connection is secure** then click the icon for **Show certificate** (upper right corner).
4. In **Certificate Viewer** dialog box, go to **Details** tab and click **Export**. Now you have the certificate that you can upload in SAP.

Configuring the ABAP Package

When using the ABAP Package to send label printing requests to Loftware Automation, you must provide the mandatory parameters with each request. Typically, these parameters are label name, printer name, label object values, destination at which Loftware Automation listens for requests, etc.

Some parameters, for example, label object values, are unique for each request. You must send unique parameters with each request. On the other hand, some parameters are the same for each request. An example of such a static parameter is the Loftware Automation destination (Web Service or HTTP request).

You can save these static parameters in a configuration table. If the requests from your SAP application do not provide any values for static parameters, the ABAP Package uses the default values from your configuration table.

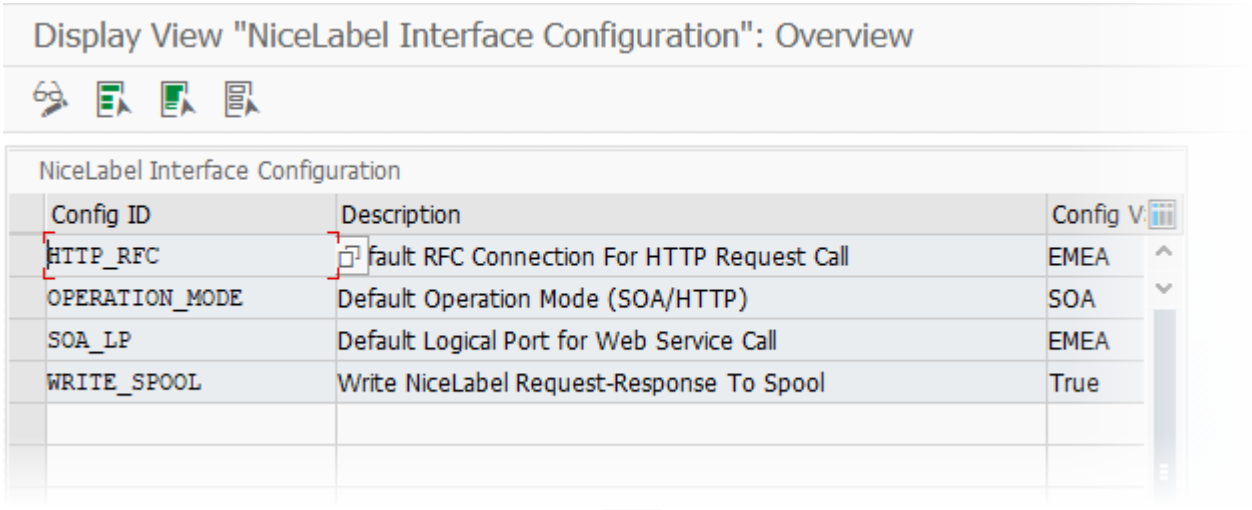
Configuring system-wide defaults

ABAP Package uses the configuration table for label printing defaults. These printing defaults include printer name, label name, default SOA (Web Service), and RFC type G (HTTP) destinations. If you do not provide these parameters in the request from your ABAP code, the ABAP package uses the defaults from the configuration table.

NOTE: This is an optional configuration. You do not have to use this table. The table can remain empty if you intend to provide the required parameters with each request.

To change the system-wide defaults:

1. In **SAP GUI**, run the transaction **/NICELAB/IF_CTRL**.
   The display view of the **Loftware Interface Configuration** table appears.
2. By default, the table is empty.
   You should include value at least for OPERATION_MODE and then one of HTTP_RFC of SOA_LP (the communication mode you intend to use).



3. To set the defaults, click **Display -> Change** button, make changes, and click **Save**.

The following configurable parameters (Config ID) are available:

CONFIG ID	DESCRIPTION
OPERATION_MODE	Defines the default communication mode towards Loftware Automation. Possible values are **SOA** (to use Web Service communication) or **HTTP** (to use RFC type G communication).
SOA_LP	Defines the default logical port for the SOA (Web Service) call. Type the name of the logical port you have previously defined in the transaction **SOAMANAGER**.
HTTP_RFC	Defines the default RFC type G (HTTP) destination. Type the name of the HTTP Connection to the External Server (type G) that you have previously defined in the transaction **SM59**.
API-VERSION	Defines the Cloud trigger API version. This is sent as a custom header in the HTTP call to Cloud trigger API.
OCP-APIM-SUBSCRIPTION-KEY	Defines the Cloud trigger subscription key. The key authorizes SAP to execute the HTTP call to the Cloud trigger. You need this key to: • Get a list of available cloud-connected printer • Print to cloud-connected printers • Get a list of label templates from the DMS. Obtaining this key is self-service action in the management of your Loftware Cloud subscription (see a chapter **Generating a subscription key to consume Loftware Cloud APIs**).
PRINTER_STATUS	Specifies if ABAP Package should send the request for the printer live status or not. Possible values are **True** or **False**.
WRITE_SPOOL	Specifies if ABAP Package logs the communication details received from Loftware Automation in the SAP Spooler. The possible values are **True** or **False**.
CATALOG_ROOT	Defines the folder in your DMS which contains labels and subfolders with labels. If no value is provided, the catalog will be created from the root folder in the DMS.
DATA_MODEL	Specifies if ABAP Package should send the request for the data model file (Field Catalog). Possible values are **True** or **False**. Use this while you design the label templates in DEV. You only need to execute it once per a different Field Catalog. Do not enable it on PROD, you don't need this functionality in production.

NOTE: The "True" and "False" values are case-sensitive so make sure to provide them correctly (with the capital initial).

Configuring transaction (process) defaults

Configuration table for transaction defaults is a part of the ABAP Package. Use this table to specify the default values for basic parameters. These basic parameters depend on the calling application (transaction code).

The print programs for supported standard SAP transactions that come with ABAP Package also use the defaults from this configuration table. For more information about the supported standard transactions and provided print programs, see chapter **Configuring transaction to call ABAP Package** on page **74**.

The purpose of the configuration table is to simplify the label printing process. Because of the configuration table, users do not need to know the exact names of certain technical parameters, such as label or printer names used on the Loftware Automation server.

If the calling application does not provide specific parameters, ABAP Package uses the values for those specific fields from the configuration table (provided that the default value is configured).

The currently supported parameters per process are label name, printer name, and label preview. These options can be defined per user. In this case, each user can print to his printer without affecting the system for other users. The alternative option is to define these options as a system-wide default for all users.

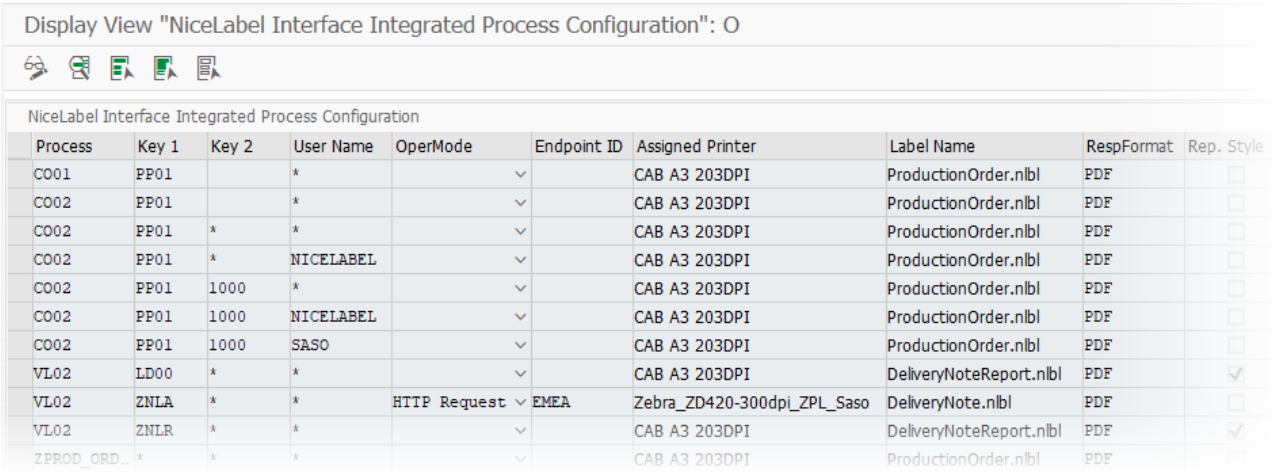
NOTE: When you define the username as an asterisk "*" the ABAP Package applies the settings for all SAP users without individually defined defaults.

The instructions below provide an example configuration for the outbound delivery process (transaction VL02). In this configuration, the processing of **output** triggers the label print:

1. In **SAP GUI**, run the transaction **SM31**.
   The window **Maintain Table Views: Initial Screen** appears.
   In **Table/View**, type **/NICELAB/V_IF_PR**.

   NOTE: You can also run a transaction /NICELAB/IF_PROC to open the table view.

2. Click **Display**.
   The **Loftware Interface: Integrated Process Configuration** details window appears. By default, the table is empty.


Display View "NiceLabel Interface Integrated Process Configuration": O

NiceLabel Interface Integrated Process Configuration

Process	Key 1	Key 2	User Name	OperMode	Endpoint ID	Assigned Printer	Label Name	RespFormat	Rep. Style
CO01	PP01		*			CAB A3 203DPI	ProductionOrder.nlbl	PDF	
CO02	PP01		*			CAB A3 203DPI	ProductionOrder.nlbl	PDF	
CO02	PP01	*	*			CAB A3 203DPI	ProductionOrder.nlbl	PDF	
CO02	PP01	*	NICELABEL			CAB A3 203DPI	ProductionOrder.nlbl	PDF	
CO02	PP01	1000	*			CAB A3 203DPI	ProductionOrder.nlbl	PDF	
CO02	PP01	1000	NICELABEL			CAB A3 203DPI	ProductionOrder.nlbl	PDF	
CO02	PP01	1000	SASO			CAB A3 203DPI	ProductionOrder.nlbl	PDF	
VL02	LD00	*	*			CAB A3 203DPI	DeliveryNoteReport.nlbl	PDF	✓
VL02	ZNLA	*	*	HTTP Request	EMEA	Zebra_ZD420-300dpi_ZPL_Saso	DeliveryNote.nlbl	PDF	
VL02	ZNLR	*	*			CAB A3 203DPI	DeliveryNoteReport.nlbl	PDF	✓
ZPROD_ORD...	*	*	*			CAB A3 203DPI	ProductionOrder.nlbl	PDF	

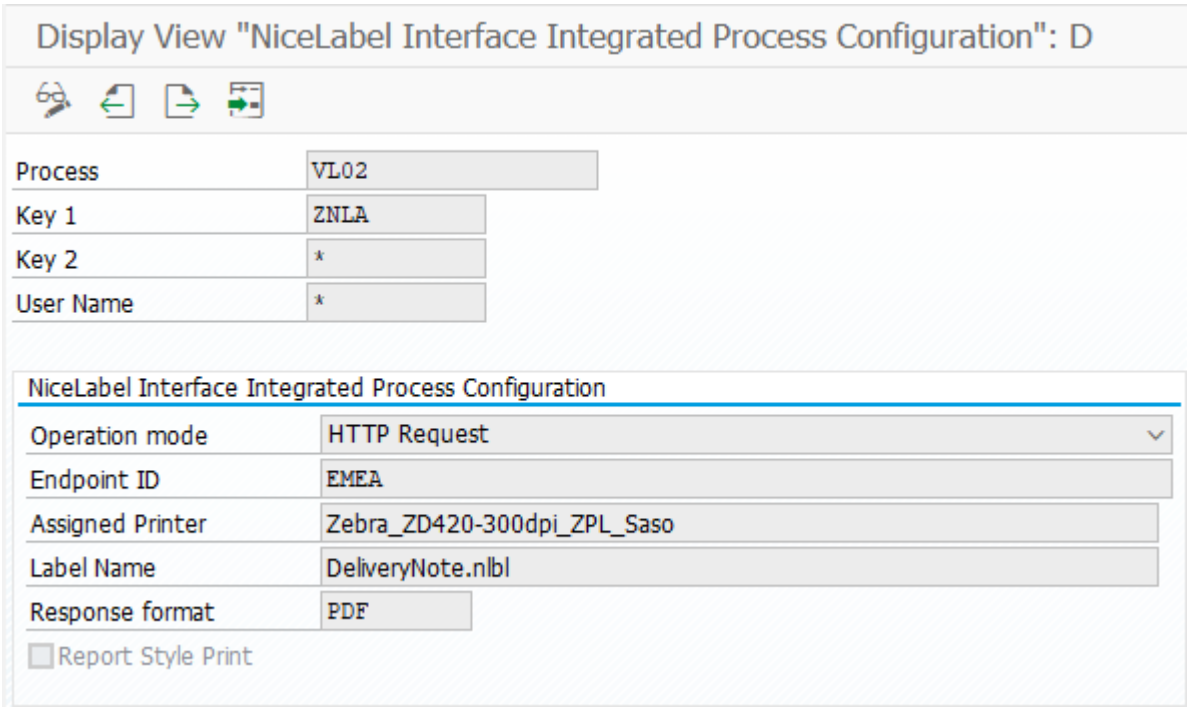
3. Click **Display -> Change** button.
4. To add a new entry in the table, click **New entries**.
   a. For **Process,** type **VL02**.
   b. For **Key1**, type **ZNLA**. This field defines the **Output Type** that you select in Outbound Delivery.
   c. For **Key2**, type ***** (asterisk). This field is not used for Outbound Delivery.
   d. For **User Name,** type the name of the SAP user for whom you want to define the default settings (you can type * (asterisk) and the rule will apply to all users).
   e. For **OperMode**, select the operation mode (the type of connection to the Automation server that you have configured for your system).
   f. For **Endpoint ID**, type the name of the defined endpoint you have configured in your system.
   g. For **Assigned Printer,** select the name of the printer available on the respective Loftware Automation system. If the list is empty, you have to send a request for a list of printers on the target Automation system. The easiest method is to click **Get Printers** button in the provided demo transaction (/NICELAB/IF_DEMO).
   h. For **Label Name,** type the name of the label to print.
   i. For **Response format,** type the format of the label preview to receive (PDF, PNG, JPEG). This setting is used, when you request a label preview.

j. For **Report style**, enable the feature if you want to print items in a report-style document (A4 or Letter page size). In this case, ABAP Package encapsulates the iterable data for the report. In most cases, you will have this field set to **False**.

For empty fields use an asterisk as value - “*”.

5. Click **Save** or press **Ctrl+S**.

This is an example of the item configured in the Process Configuration table:



In the above example, when ABAP Package receives the label printing request from Outbound Delivery (tcode VL02N) for any user and the selected Output Type is **ZNLA,** the label **`DeliveryNote.nlbl`** prints to printer **`Zebra_ZD420-300dpi_ZPL_Saso`**. Because **Report Style Print** is false, each item from the delivery will print on a separate label (you will receive as many labels as there are items in a delivery). The print request is sent as RFC request (HTTP Request) to RFC destination **EMEA** (as defined in tcode SM59).

In the above example, the criteria to pick up default print settings are a transaction (process), key 1, and user. If you need additional criteria (like document type, company code, plant, etc.) you can use the remaining generic Key field (Key2) also available in the table, to maintain the configuration. However, if you decide to use additional Key fields, you also must adapt your ABAP coding to check for the generic key columns.

The configuration table is used in the print programs delivered with the ABAP Package. You can also use the table for the print programs you have developed yourself for other transactions.

Structure of “Integrated Process Configuration” table

See the structure **/NICELAB/V_IF_PR**.

The table contains business rules configured for label printing from SAP transactions. ABAP Package uses this table to apply rules in the print programs provided with the ABAP Package for the supported transactions.

You can also use this table to configure print rules from your custom print programs.

FIELD NAME	FIELD DESCRIPTION
Process	Specifies the process name from which print request is initiated and that will trigger the rule processing.
Key 1	The custom key that you can use in your print programs to create business rules. In the provided print programs, the key is used as follows: • /NICELAB/OUTPUT_PROCESSING (for VL02n): the key is used to define the **output type** for outbound delivery the **Output Type** (e.g., LD00). • /NICELAB/PSFC_OBJECT_LIST (for CO02): the key is used to define the **order type** for production orders.
Key 2	The custom key that you can use in your print programs to create business rules. In the provided print programs, the key is used as follows: • /NICELAB/OUTPUT_PROCESSING (for VL02n): not used. Use * (the asterisk) as the value. • /NICELAB/PSFC_OBJECT_LIST (for CO02): the key is used to define the **plant**.
User Name	Specifies the name of the user to whom the rule applies. Use * (asterisk) to apply a rule to all users.
OperMode	Specifies the operation mode – how will ABAP Package communicate with the Automation backend. The possible options are: • **SOAP Proxy Call**: Your Automation endpoint is accessible via a SOAP call. • **HTTP Request**: Your Automation endpoint is accessible via an HTTP REST call. • **SAP PI** (Cloud Trigger Endpoint): Your Automation endpoint is accessible via an HTTP REST call through PI/PO infrastructure. • **SAP PI**: Your Automation endpoint is accessible via a SOAP call through PI/PO infrastructure. • **SAP Cloud Platform Integration**: Your Automation endpoint is accessible via a SOAP call through SAP CPI infrastructure (routed to Automation's HTTP REST endpoint – Cloud trigger).
Endpoint ID	Specifies the ID for your endpoint. For example, when you select SOAP Proxy Call, this would be the **Logical Port** defined in tcode **SOAMANAGER**. When you select HTTP Request, this would be the **RFC Destination** defined in tcode **SM59**.
Assigned Printer	Specified the printer queue name as available from the selected OperMode/Endpoint ID destination.
Label Name	Specifies the label template name that you want to print.
RespFormat	Specifies the format of the label preview, if you have requested a label preview. The allowable values are **PDF**, **PNG**, and **JPEG**.
Rep. Style	Specifies whether you will provide the values for a report-style output. For example, this would be A4 or Letter label template with multiple items printed on the same page for the shipment documentation.

Enabling enhancement spots

For more information about what are enhancement spots, see chapter **Enhancement spots** on page **74**.

Configuration table for enhancement spots is a part of the ABAP Package. Use this table to enable enhancement spots for the supported standard SAP applications.

When enhancement spots are enabled and users produce standard outputs in SAP, the enhancement spots execute a code to seamlessly send the transaction data and commands into ABAP Package. The Loftware Automation backend prints the transaction data on the associated label.

To enable the enhancement spot for SAP transactions:

1. In SAP GUI, run the transaction **/NICELAB/IF_ENHS**.
   The list of available enhancement spots will display.
2. Click **Display -> Change**  button.
3. Configure details for each displayed enhancement ID.
   a. **Enh. Active.** Specifies whether the enhancement spot for the selected output is active.
   b. **Assigned Printer.** Specifies the name of the printer driver that prints labels. This is the name of the Windows printer driver installed on the computer with Loftware Automation.
   c. **Label Name.** Specifies the label name that prints for the selected output. For more information on the label names, see chapter **Providing a label** name on page 67.
   d. **RespFormat.** Specifies the graphical format for the label preview.

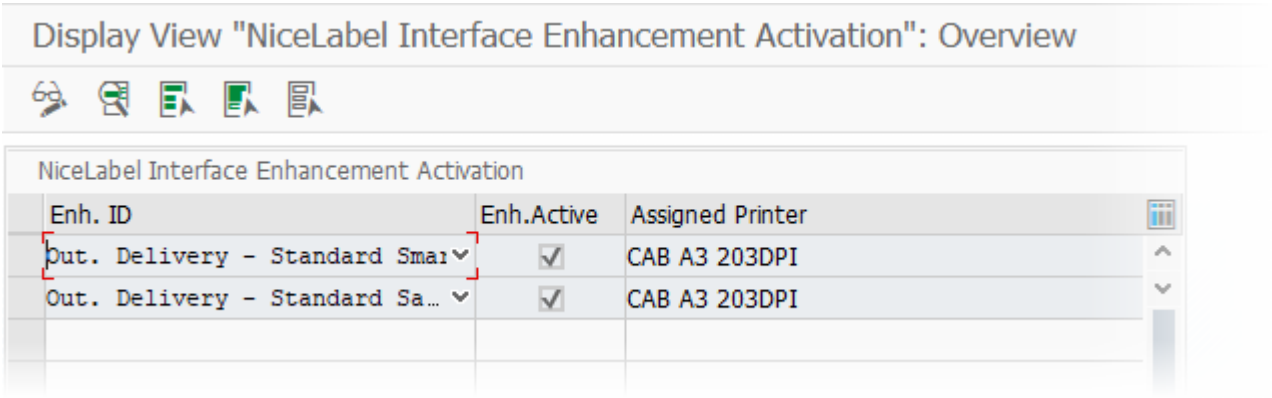


Figure 1: List of available transactions with enabled enhancement spots

Enabling built-in output devices in SPAD

NOTE: Complete the steps in this chapter if you intend to log communication with the Loftware integration system to SAP Spooler.

ABAP Package also transports the output devices that are associated with entries in SAP Spooler after enabling the **Write to Spool** functionality.

Each SAP Spooler entry is associated with an output device. ABAP Package transports two output devices:

- **NiceLabel Request To Spool (NLRS).** Most entries are associated with this output device.

- **NiceLabel Raw Printer (NLBN).** This output device is used for entries that store the binary print job.

After you transport the ABAP Package, the new output devices are in a locked state.

To enable the output device:

1. In **SAP GUI**, run tcode **SPAD**.
2. In **Output Devices**, type **NiceLabel Request To Spool** and press Enter.
3. Click the **Change** button  or press **F8**.
4. In **Spool Server**, select your server.
5. Click **Save** button  or press **Ctrl+S**.

Repeat the above steps also for the output device **NiceLabel Raw Printer**.

You can also associate the print job in SAP Spooler with your specific output device. See the chapter **Defining your own Output Device for print jobs in SAP Spooler** on page 65.

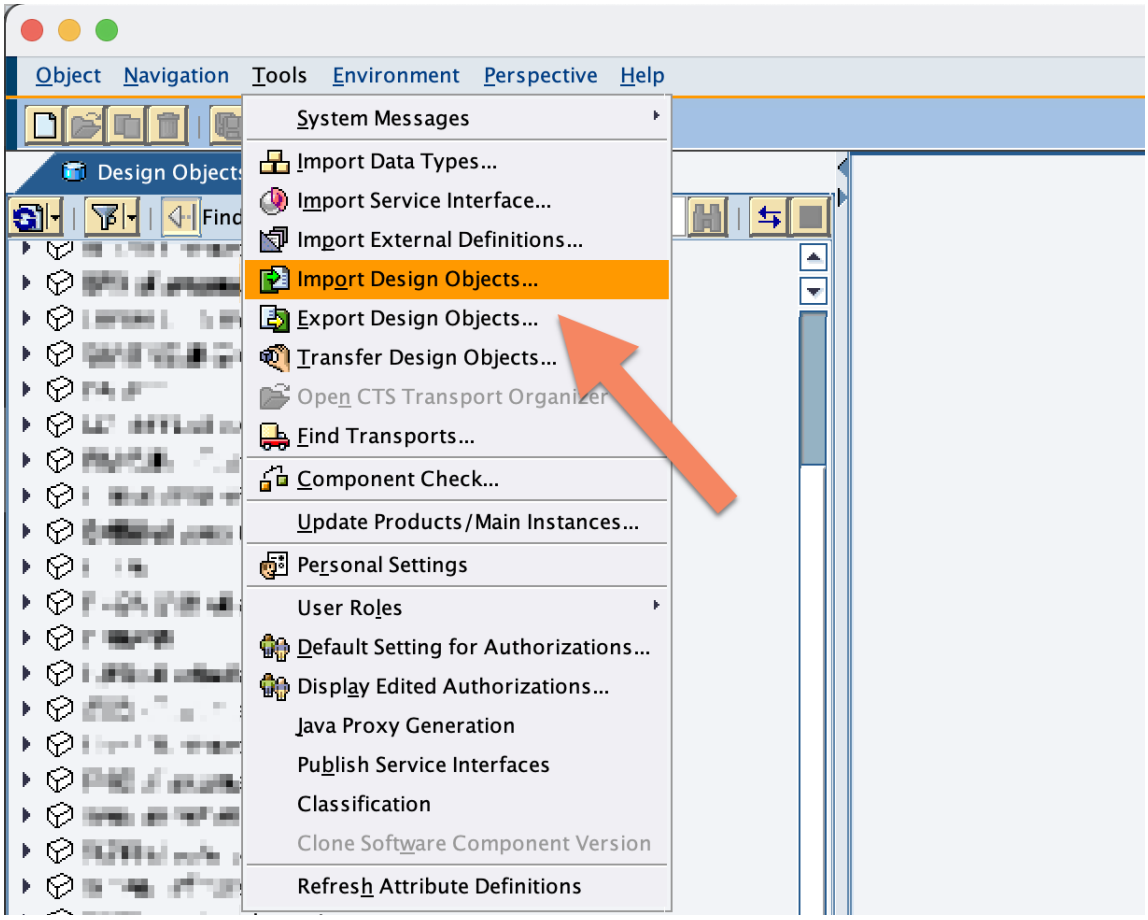
Support for PI/PO

Loftware provides PI/PO configuration that will connect to the HTTP SOAP (Web Services) endpoint or the HTTP REST (Cloud trigger) endpoint in Loftware Automation and provide the synchronous communication framework.

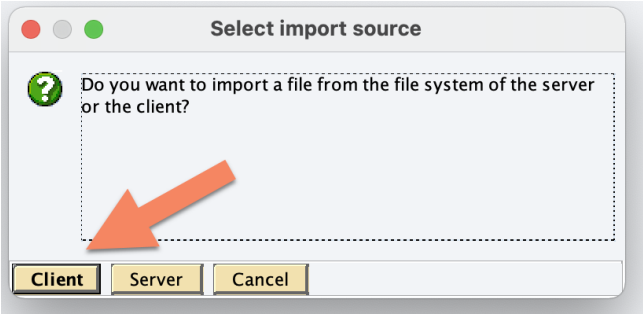
Configuring SAP PI

To configure SAP PI to accept messages from the ABAP Package and communicate them with the endpoint in Loftware Automation, do the following:

1. Log on to your **SAP NetWeaver – Process Integration console**.
2. Run **Integration Builder**.
3. In the **Tools** menu, select **Import Design Objects**.

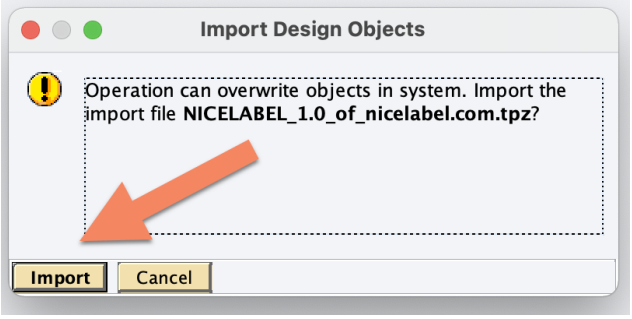


Click **Client**.

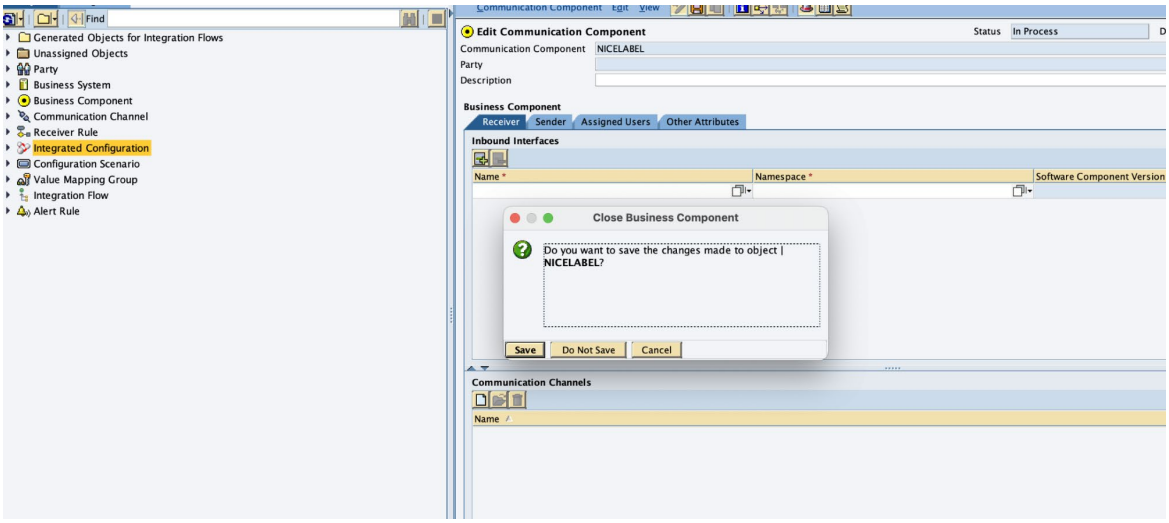


Browse for the file **NICELABEL_1.0_of_nicelabel.com.tpz** included in the ABAP Package bundle. See the folder \Integrations\PI-PO Configuration.

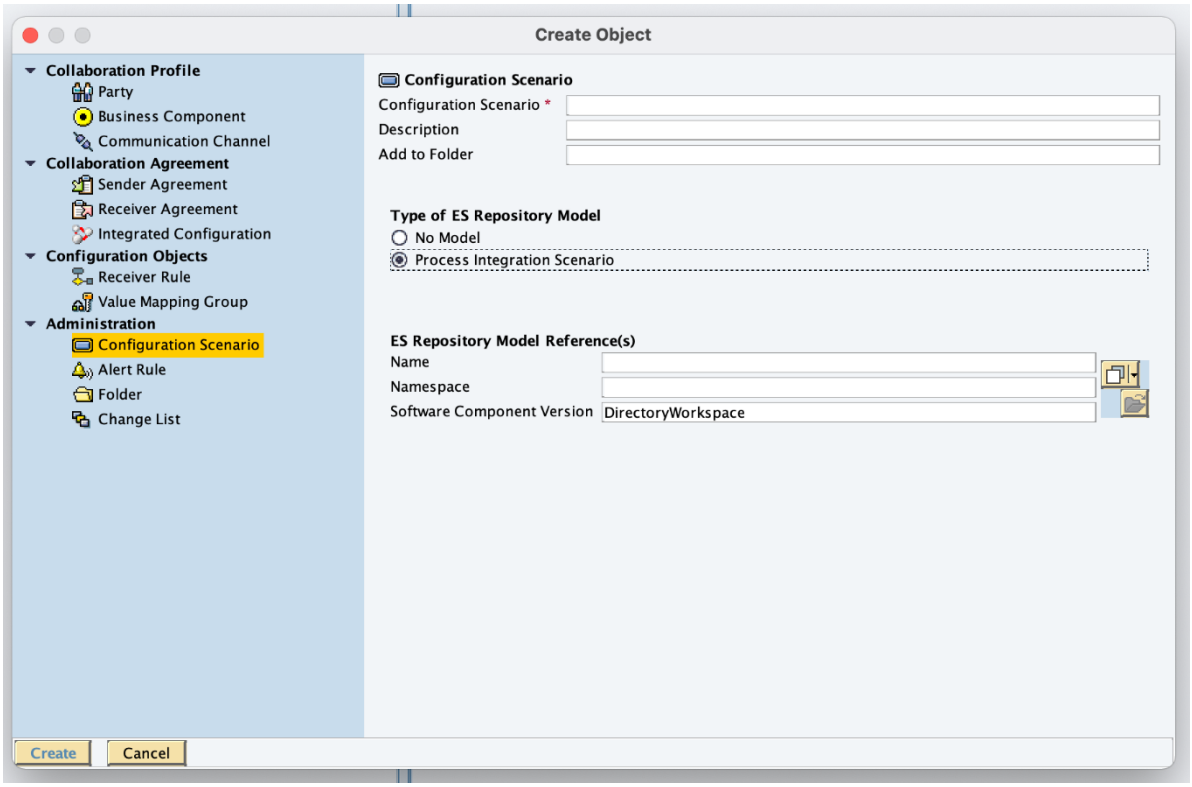
Click **Import**.



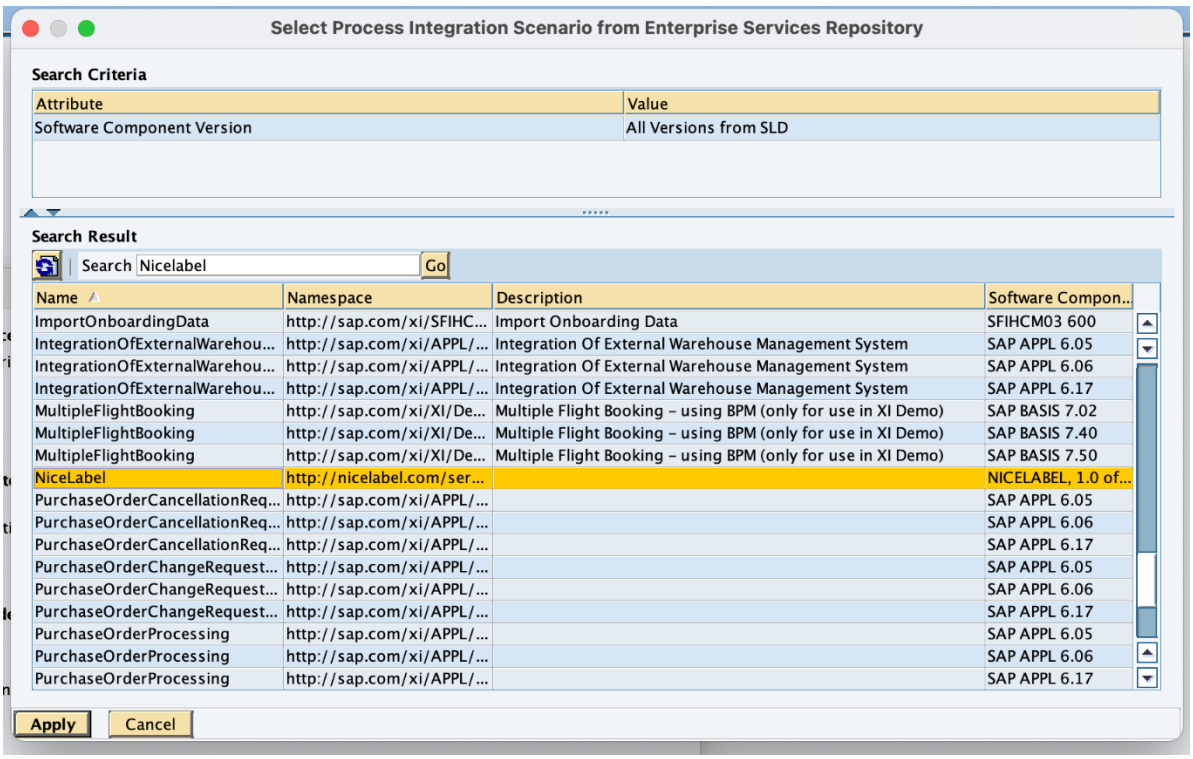
4. The next step is creating a new Business Component for your Loftware target system.
   Click **New**.
   In Create Object window, select **Business Component**. For **Communication Component**, enter "NICELABEL". You can leave **Inbound Interfaces** and **Communication Channels** empty.
   Save the component.



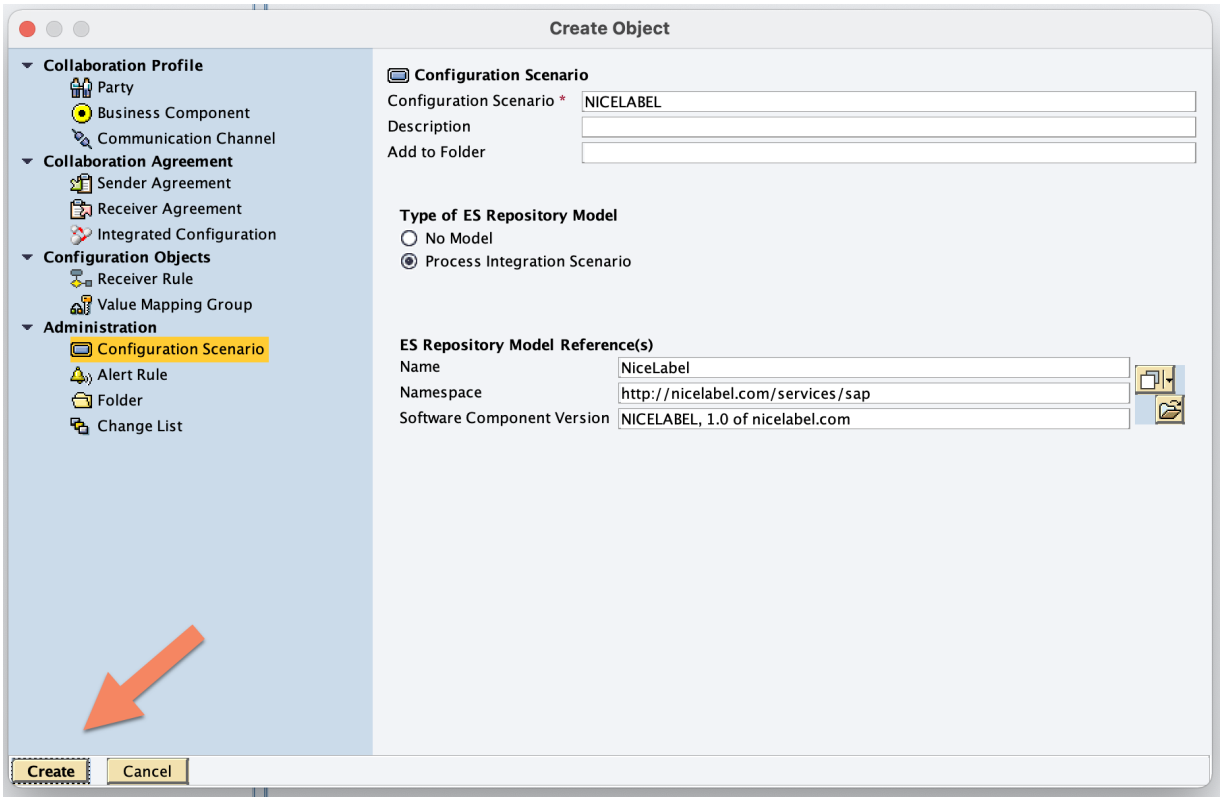
5. Create a new Configuration Scenario.
   For the Type of ED Repository Model, select **Process Integration Scenario**.



6. For the Name of the ES Repository Model Reference, browse for the **NiceLabel** scenario.

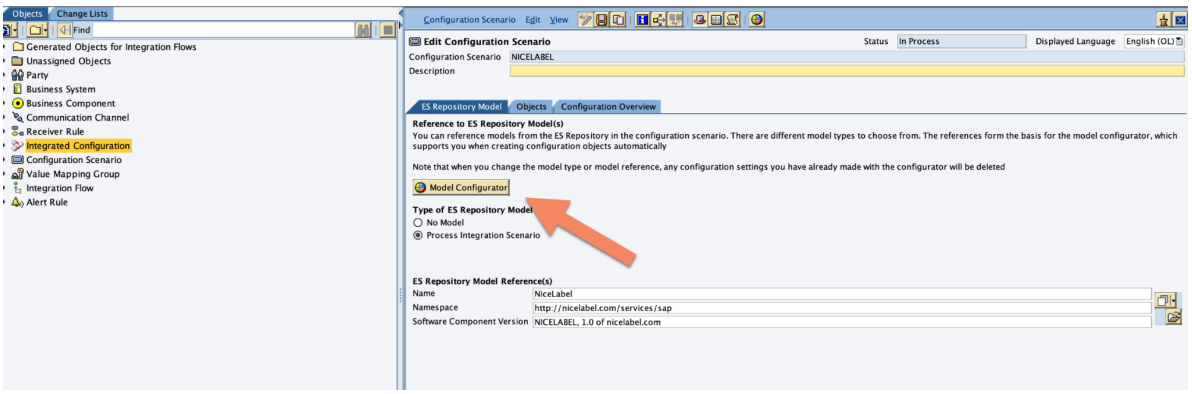


7. Click **Create** button to generate the Configuration Scenario.



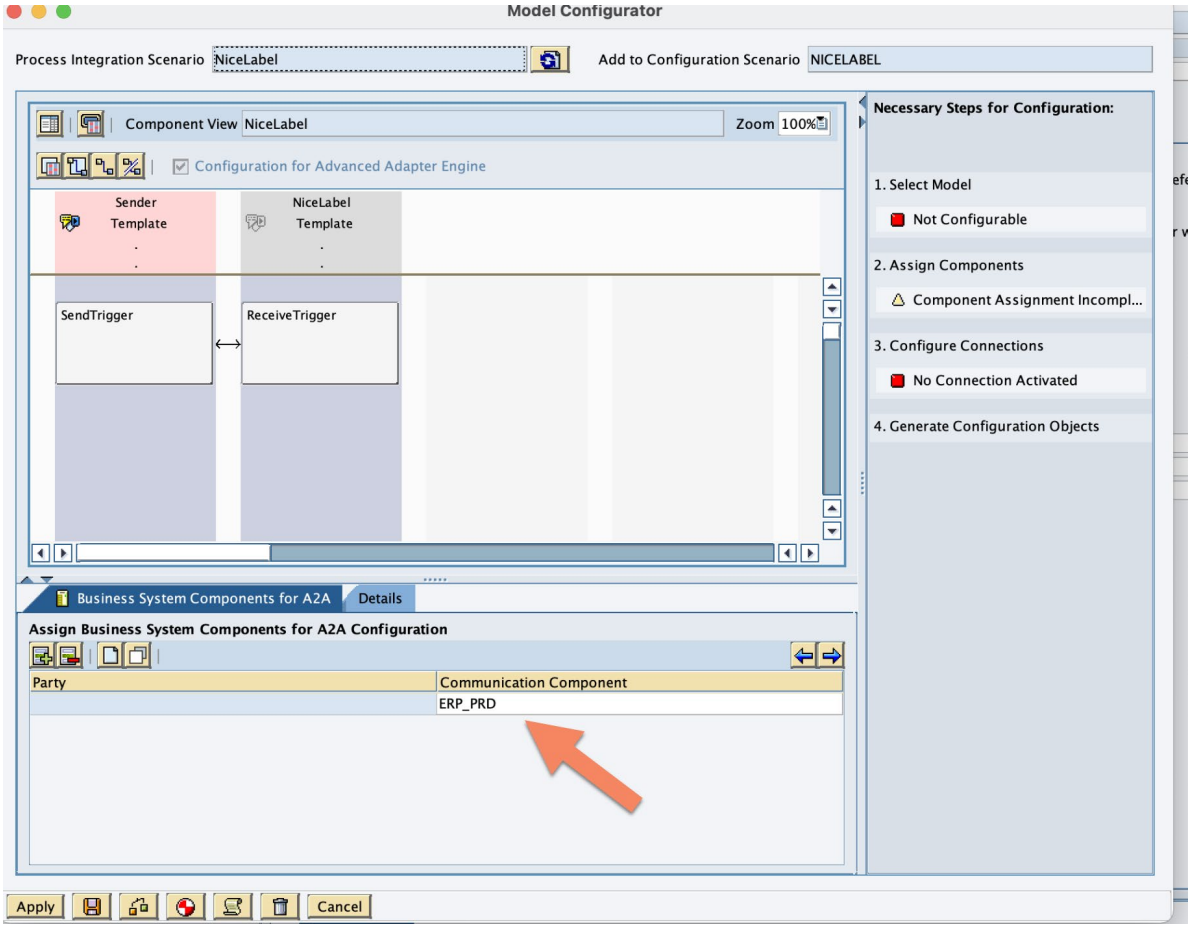
The **Edit Configuration Scenario** window will open.

8. Click **Model Configuration**.



9. Select **Sender Template**.
   For **Communication Component** in **Business System Component for A2A**, select the sender SAP system. This is a source SAP system that has installed the ABAP Package and will send data into the PI.

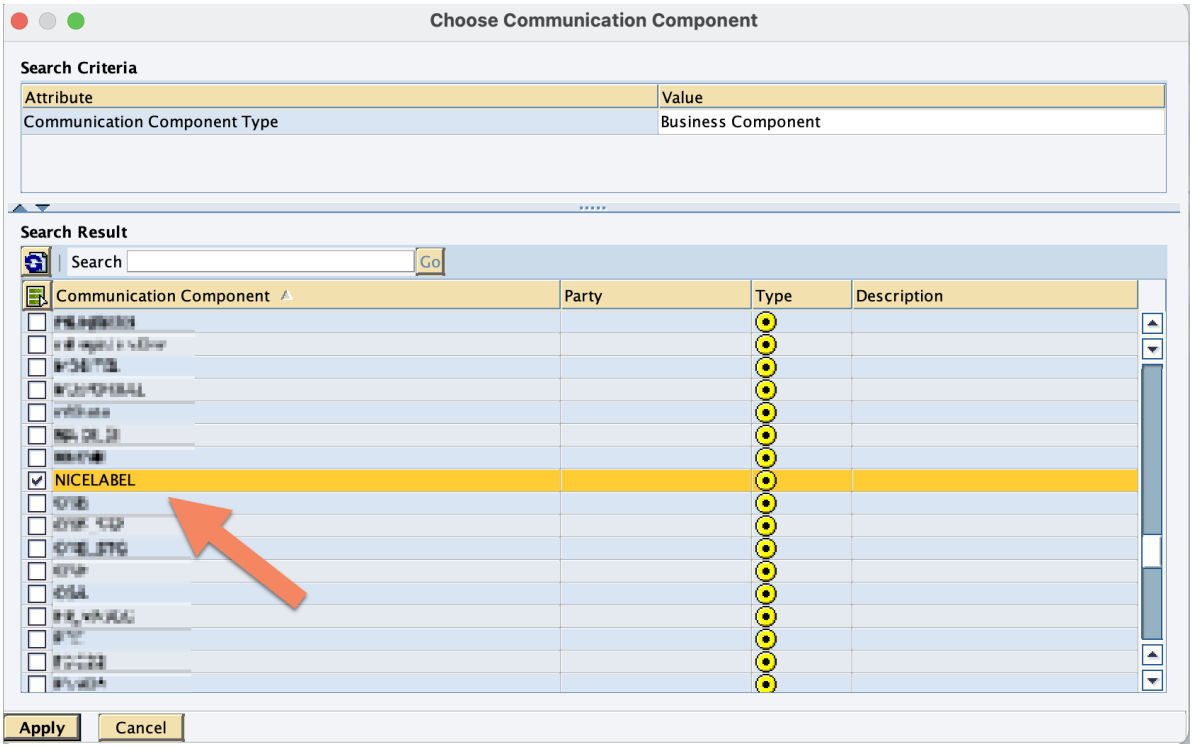
   In this example, we have selected the **ERP_PRD** system.



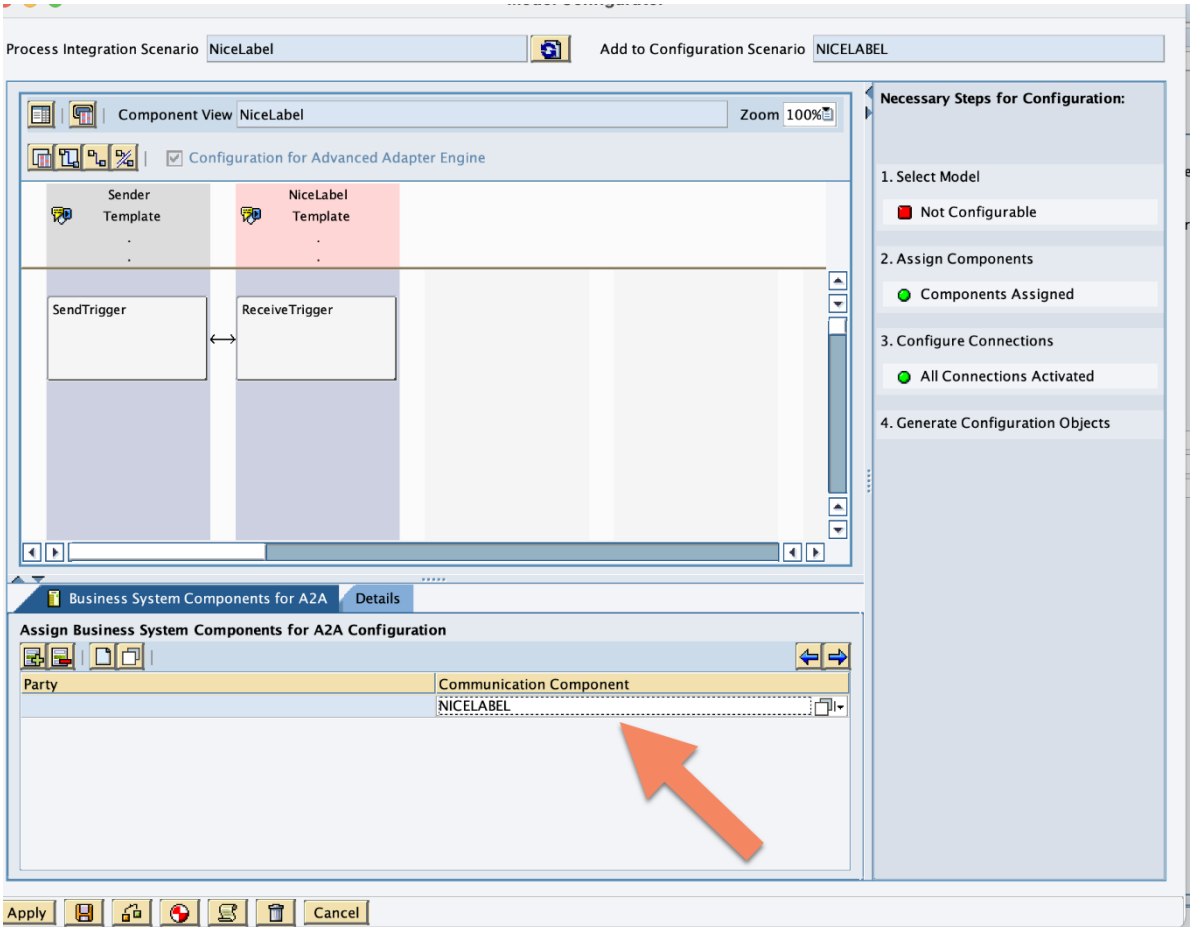
10. Select **NiceLabel Template**.
    For **Communication Component** in **Business System Component for A2A**, select the target Business System, where Loftware Automation is installed.

    In **Choose Communication Component** window, change the value of **Communication Component Type** to **Business Component**.

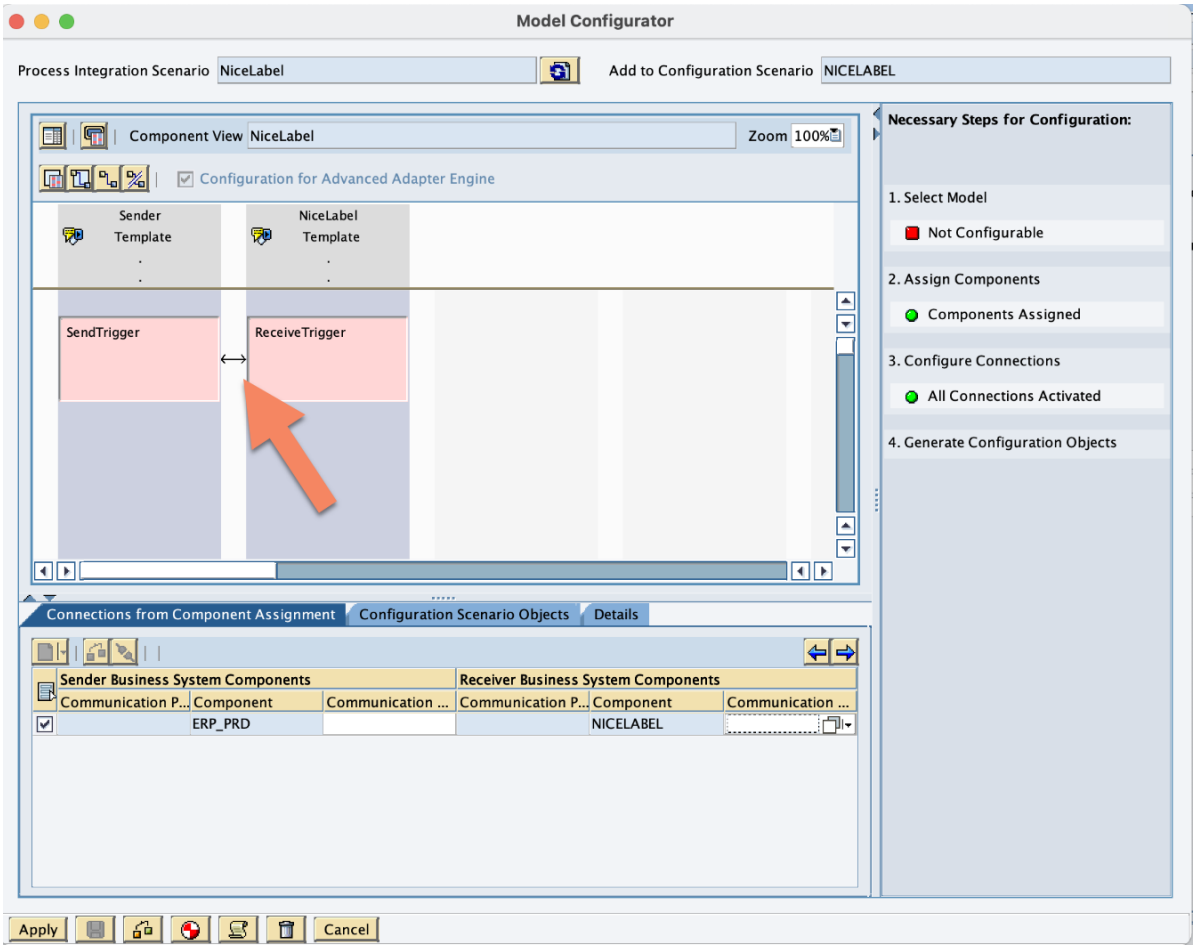
In the list of **Communication Components**, select **NICELABEL**. This is the business component you created earlier in these instructions.



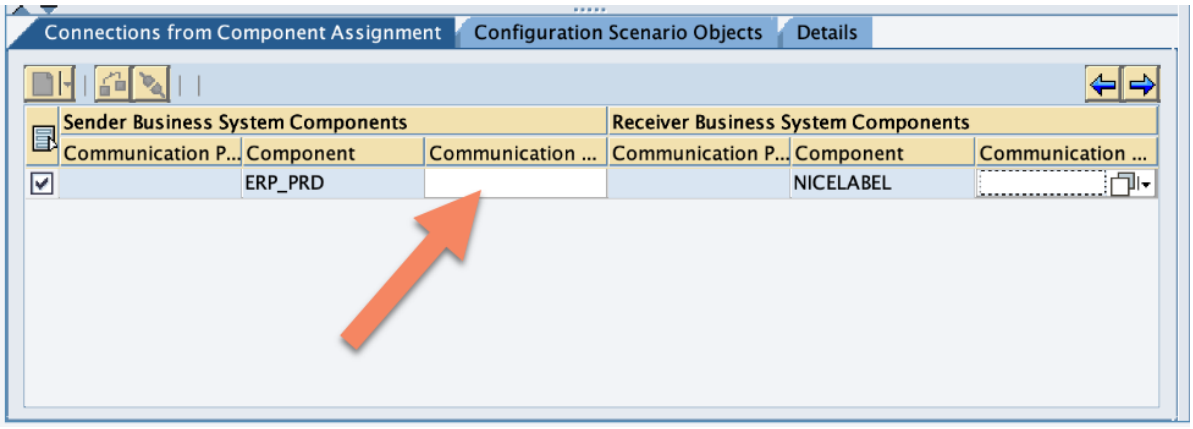
Click **Apply**.



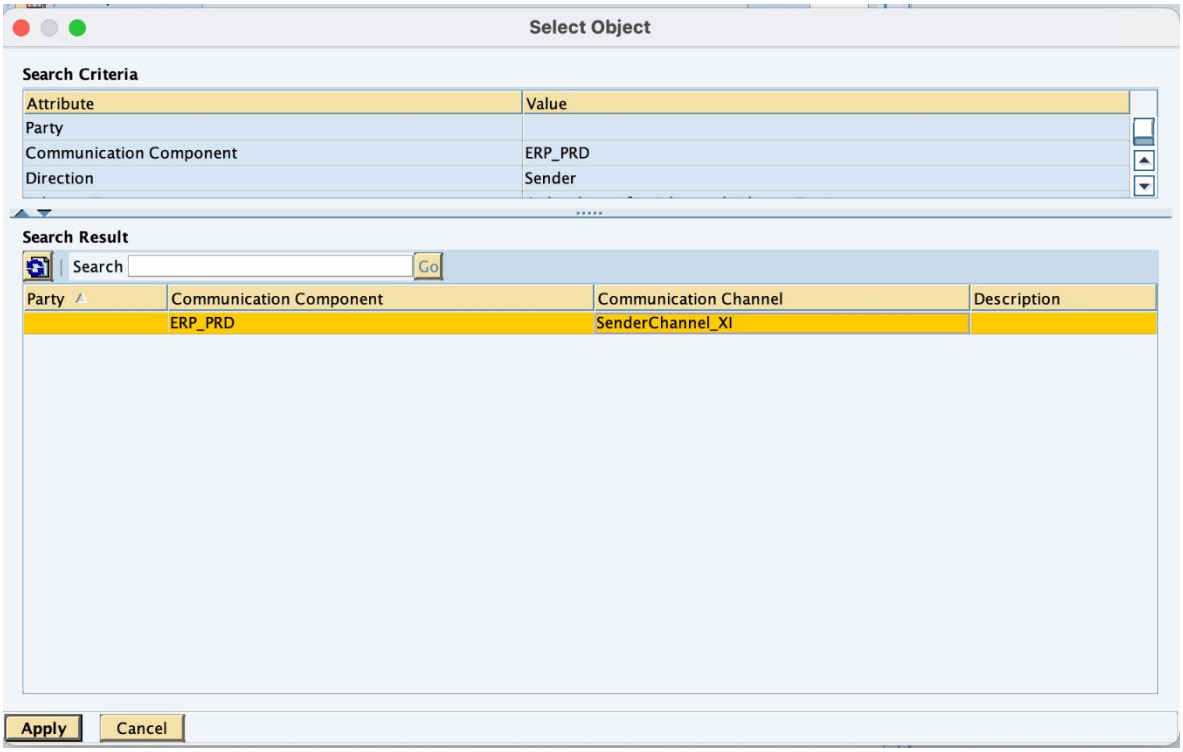
11. Click on the arrow between **SendTrigger** and **ReceivedTrigger** objects to display the **Communication Channel** fields.



12. In **Sender Business System Components**, select the **Communication Channel.**

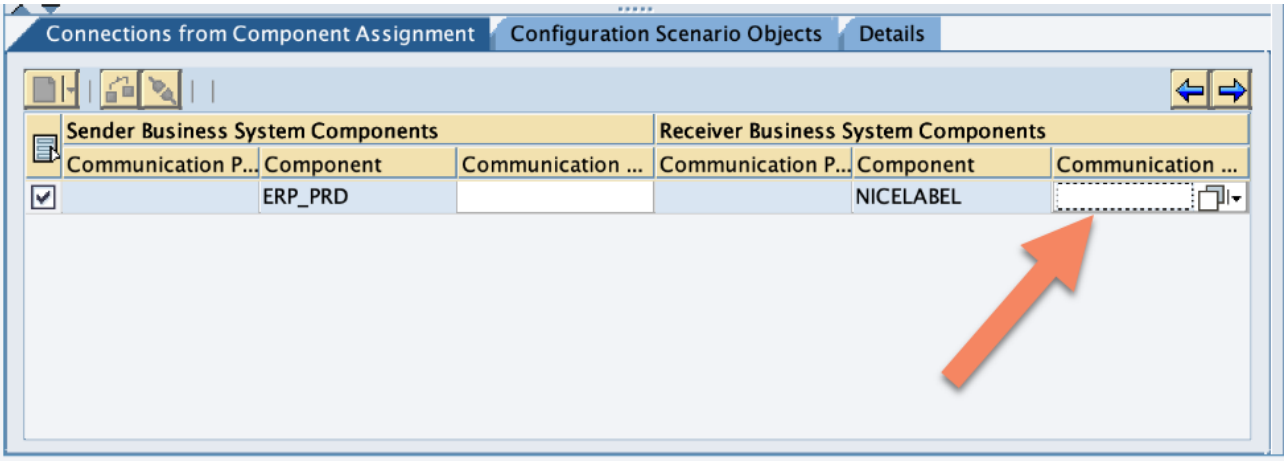


13. In the **Select Object** window, select the existing Communication Channel already defined in the source ERP system. If you have PI implemented, this Communication Channel already exists.

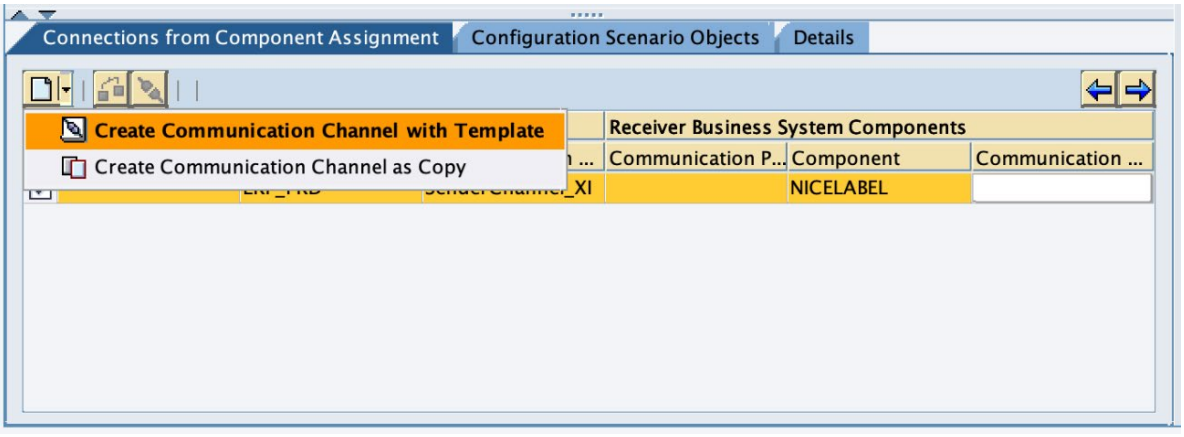


Click **Apply**.

14. In **Receiver Business System Components**, select the **Communication Channel.**

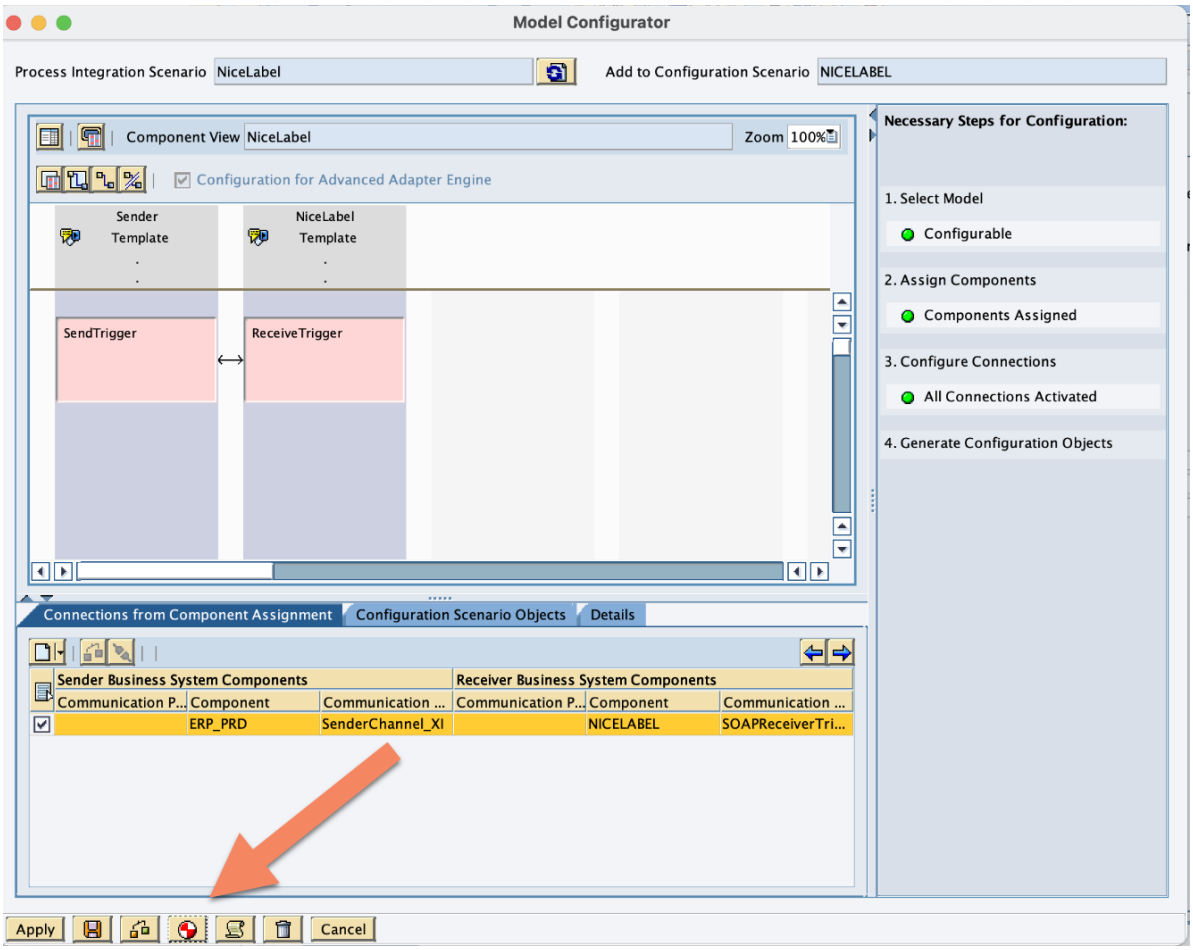


15. Click **Create Communication Channel with Template**.

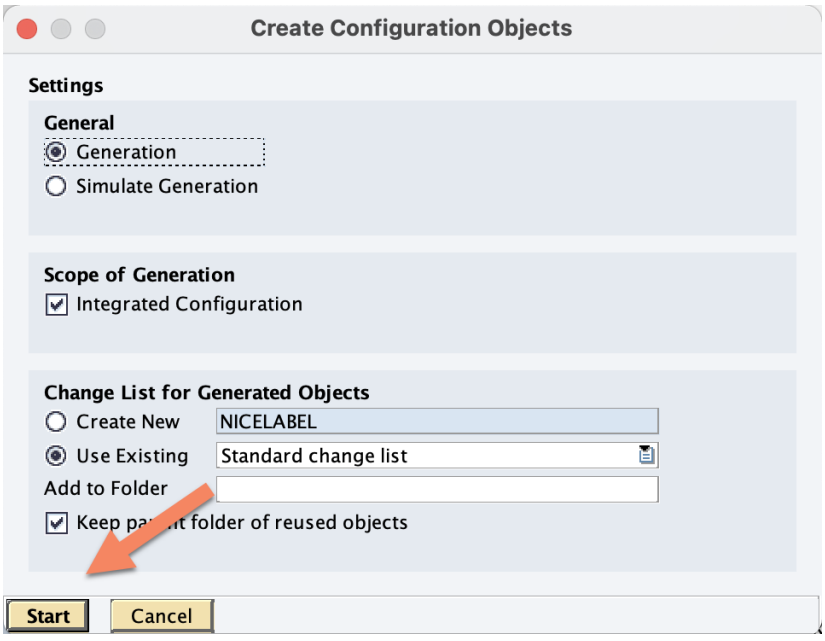


Click **Continue** through the **Create Communication Channel** steps.
Click **Finish**.

16. Click **Create Configuration Objects...**



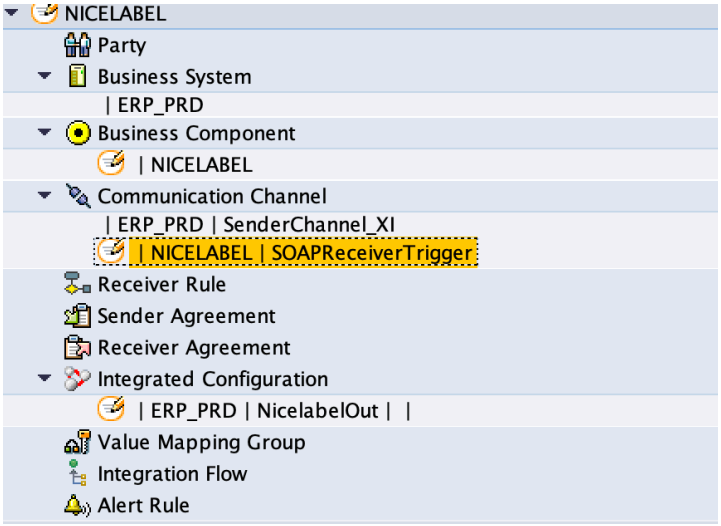
17. In **General**, select **Generation**.
Click **Start**.



After the objects have been created, close the summary screen.

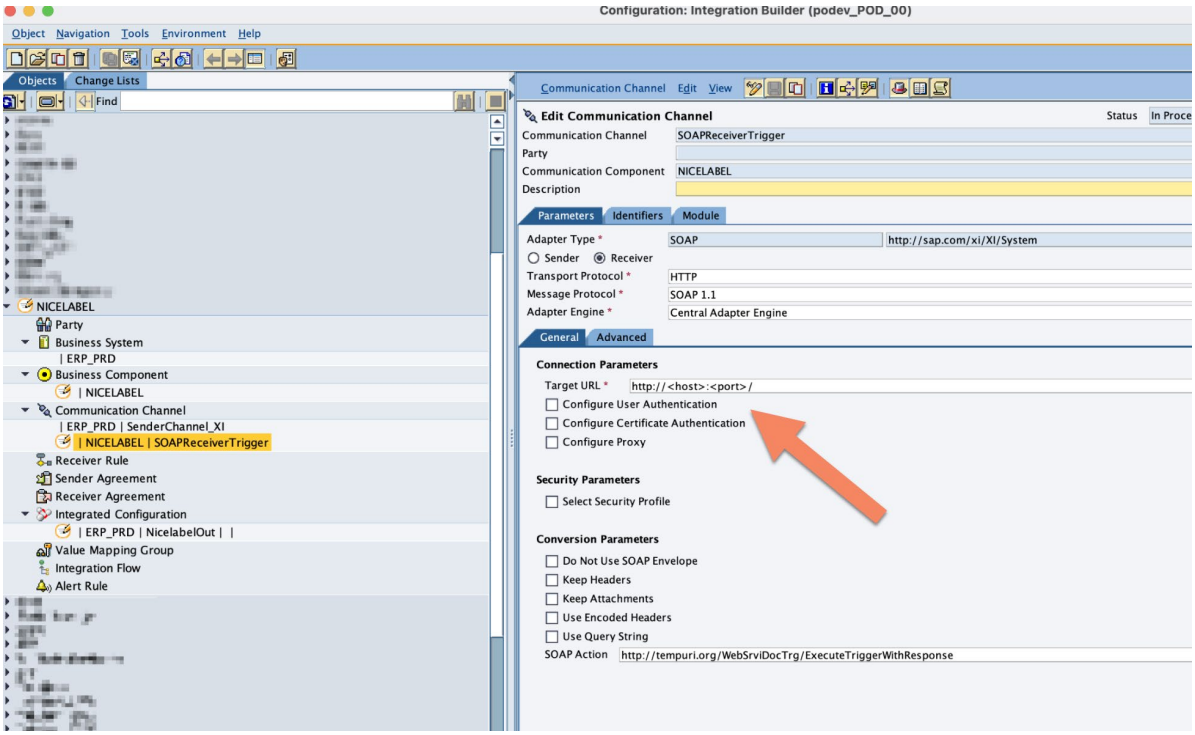
18. Click **Apply**.
19. Close the **Edit Configuration Scenario** window.
    The new configuration scenario is created.

    Select the communication channel **SOAPReceiverTrigger** (highlighted in the screenshot).
    Properties of the communication channel will display in a panel on the right.



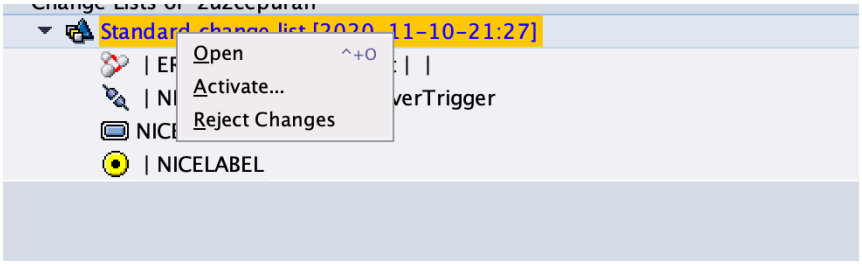
20. Click the **Change** icon to unlock editing.
    In **Connection Parameters**, enter the correct value for **Target URL** to use the server, where your Loftware Automation is installed. For <host> enter the hostname or IP address of the server, for <port> enter 50000.

    Make sure to include the **trailing slash character (/)** in the **Target URL**.

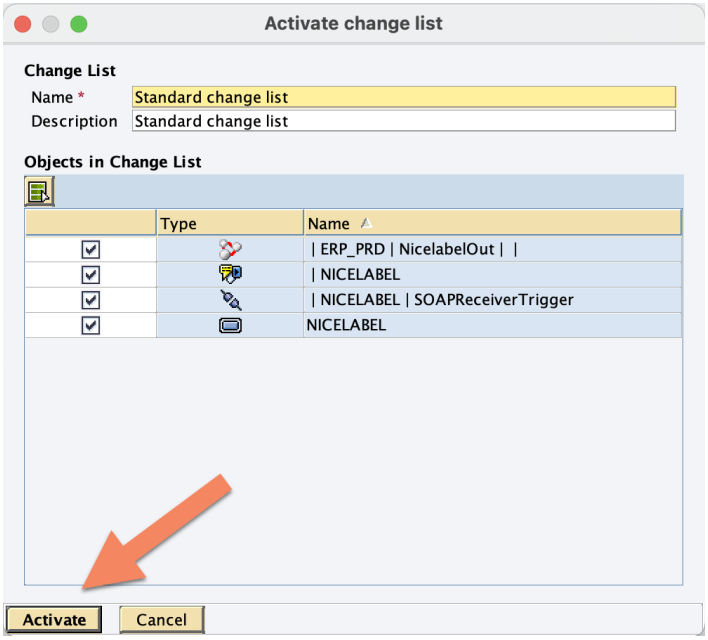


Close the **Edit Communication Channel** window.

21. Click **Change Lists** tab to display the list of changed objects.
    Right-click the **Standard change list** item and select **Activate**.



In **Activate change list**, click **Activate**.



Click **Close**.

Your PI/PO is now ready for ABAP Package to send data into it.

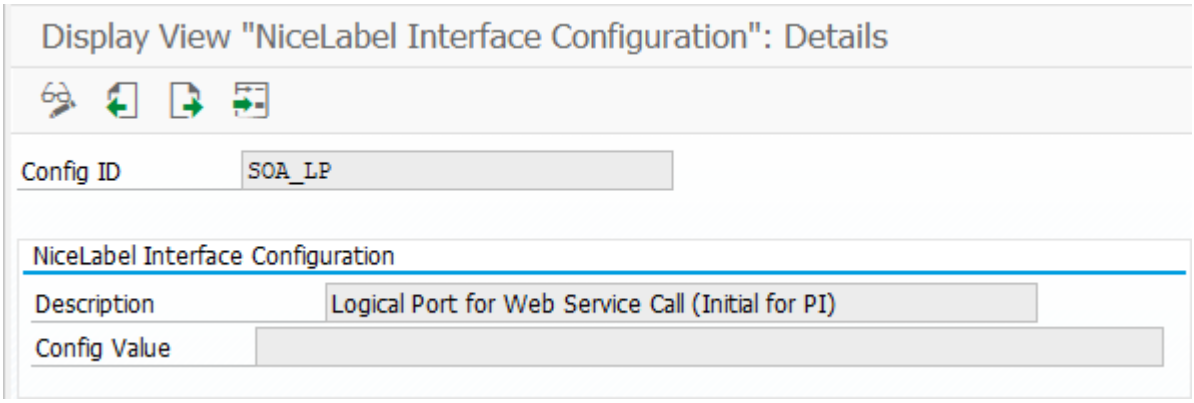
Triggering print request via SAP PI

On the SAP side, the same ABAP proxy class is used to call the Loftware web service, regardless of whether you use direct communication (SAP <-> Loftware) or communication via SAP PI (SAP <-> SAP PI <-> Loftware.

If you would like to use communication via SAP PI, then it is enough to not specify the default logical port before calling the web service (leave it empty).

- When a logical port is set, the system reads endpoint information from a logical port configuration that you defined in SOAMANAGER.
- When no logical port is set, the business system agreement configuration is looked up in your PI system.

A default logical port is set using configuration ID SOA_LP in /NICELAB/IF_CTRL transaction:



Support for Cloud Platform Integration (CPI)

To configure SAP CPI to accept messages from the ABAP Package and communicate them with the endpoint in Loftware Automation, complete steps in the following chapters.

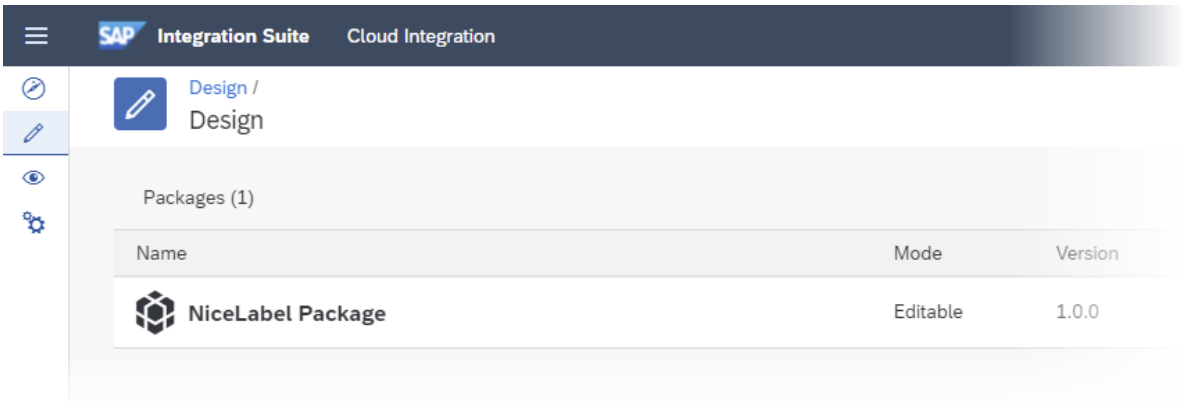
Prerequisites (not covered in this document):

- SAP user that will execute calls from ABAP Package to CPI already created.
- SAP Integration Suite subscription already set up and configured. You must have the “Design, Develop and Operate Integration Scenarios” capability available (the “Message Queues” option is not required). You must have a user login with appropriate Role Collections for editing the Integration Suite (such as PI_Integration_Developer or PI_Administrator)

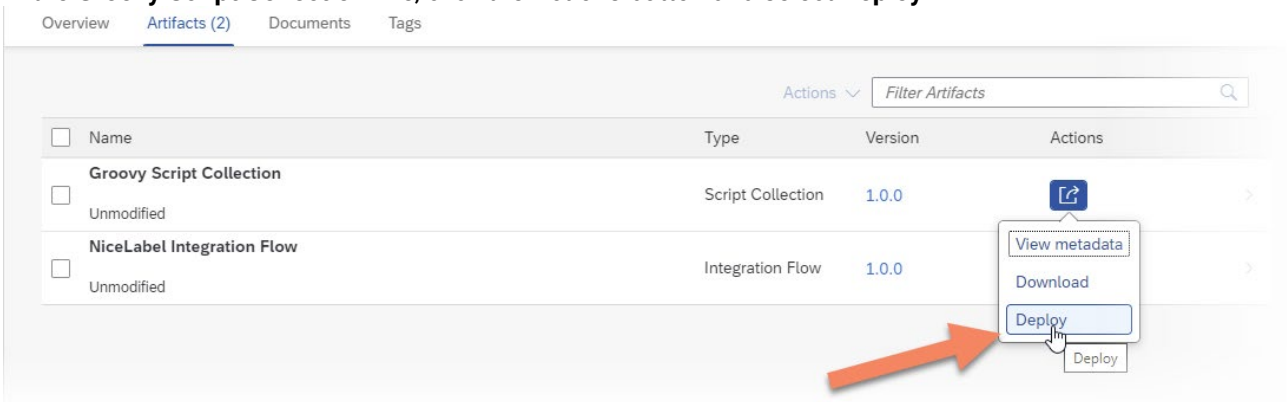
Configuring SAP Integration Suite

1. Log on to your **SAP BTP Cockpit**.
2. Click the subaccount, where you would like to configure printing through Loftware.
3. Make sure your user is a member of the appropriate Role Collection (such as PI_Integration_Developer and/or PI_Administrator) to be able to add the integration.
4. In the left-hand pane, expand the **Services** node and select **Instances and Subscriptions**.
5. Click the **Integration Suite**.
6. Select **Design, Develop and Operate Integration Scenarios**.
7. In the left-hand pane, click the **Pencil** icon.
8. Click the **Import** button.
9. Browse for the **NiceLabel Package_1.0.0.zip** file provided in the ABAP Package bundle and select it. See the folder \Integrations\CPI Configuration.

10. Select the **NiceLabel Package** you just imported.

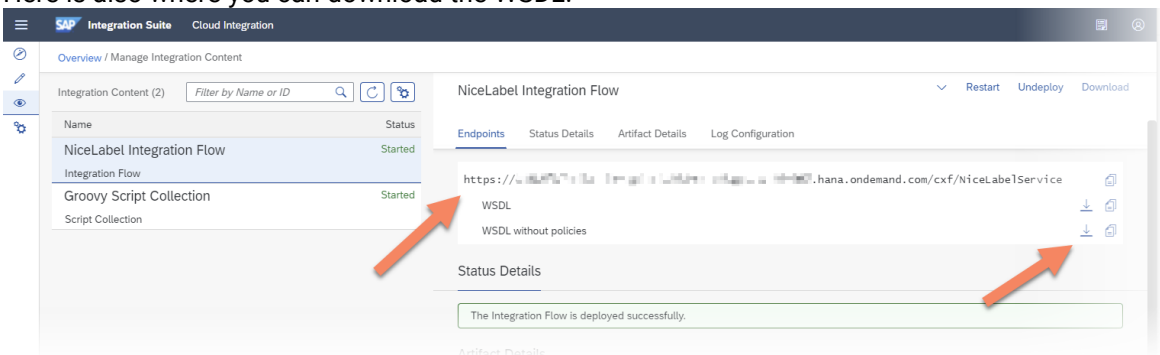


11. Go to the **Artifacts** tab.
12. In the **Groovy Script Collection** line, click the Actions button and select **Deploy**.



13. In the **NiceLabel Integration Flow** line, click the Actions button and select **Deploy**.
14. Wait until the artifacts are deployed. This might take several minutes.

    NOTE: You can track the deployment status in **Eye icon>Manage Integration Content>All**.
15. In the left-hand pane, click the **Eye** icon.
16. In **Manage Integration Content**, click on **All** tile.
17. Select **NiceLabel Integration Flow** and click the **Endpoints** tab.
    Here is where you can see the endpoint URL that you have to consume from the ABAP Package.
    Here is also where you can download the WSDL.



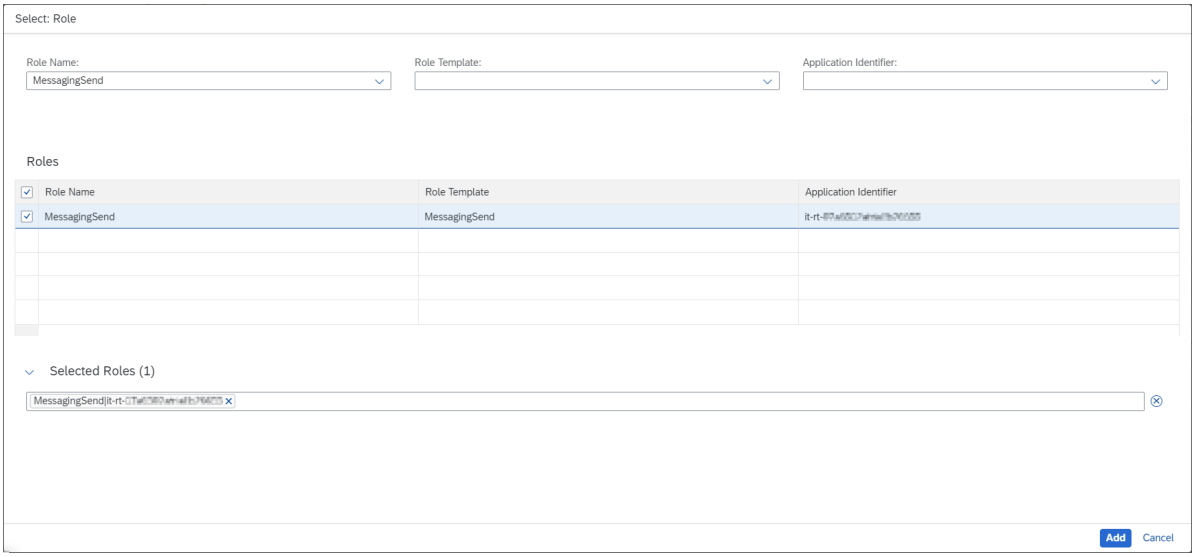
18. Click the download icon next to **WSDL without policies** to download the WSDL. You will need it later in tcode SOAMANAGER to configure the Logical Port.

    NOTE: A copy of the WSDL file is also included in the ABAP Package bundle. See the folder \Integrations\CPI Configuration.

Configuring access permissions

1. Log on to your **SAP BTP Cockpit**.
2. Click the subaccount, where you would like to configure printing through Loftware.
3. To configure access permissions, do the following:
4. In the left-hand pane, expand the **Security** node and select **Role Collection**.
5. Click the **+** button to create a new Role Collection.
6. For **Name**, type “NiceLabel_MessageSend”.
7. For **Description**, type “Allows users to send messages into CPI”.

8. Click the new role you’ve just created.
9. Click the **Edit** button.
10. Click inside the edit field **Role name**.
11. In the Role Name, find the role **MessagingSend** and select it. This will filter the view in the Roles table below.
12. Select **MessagingSend** in the Roles table. Make sure the proper Application Identifier is selected.

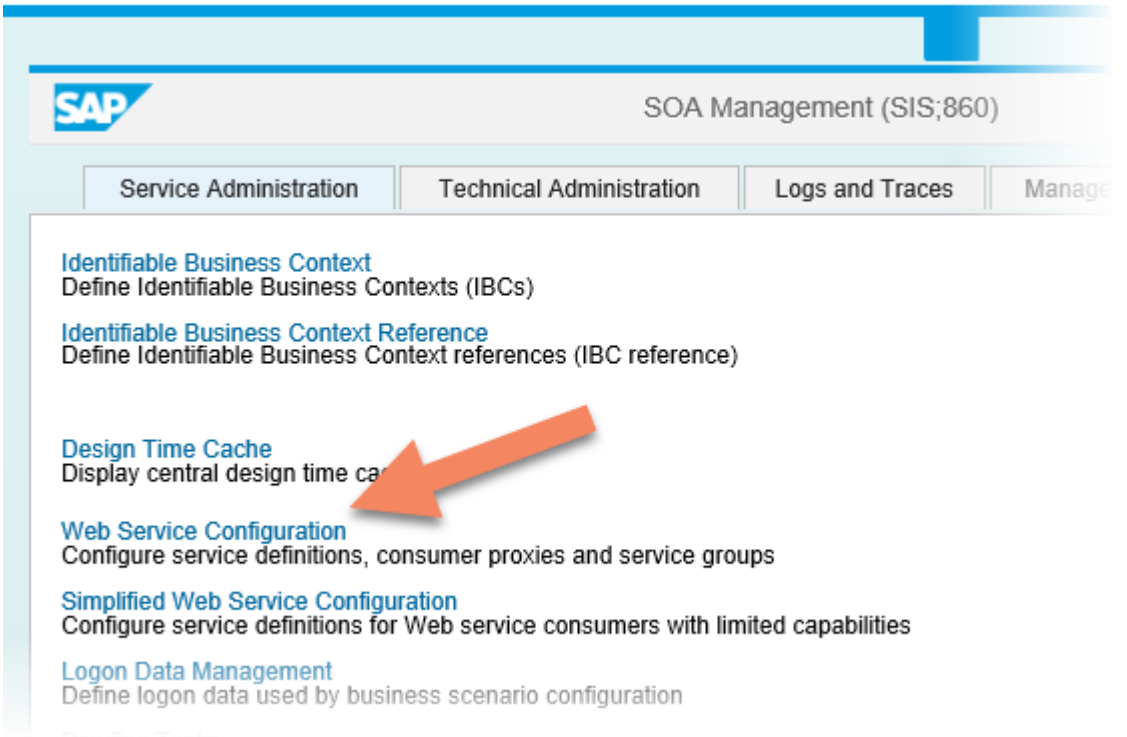


13. Click **Add**.
14. In the **Users** section, add the ID of the user that you will use for authentication. This is the user that you will have to configure in SOAMANAGER in the properties of the Logical Port.
15. Click **Save**.

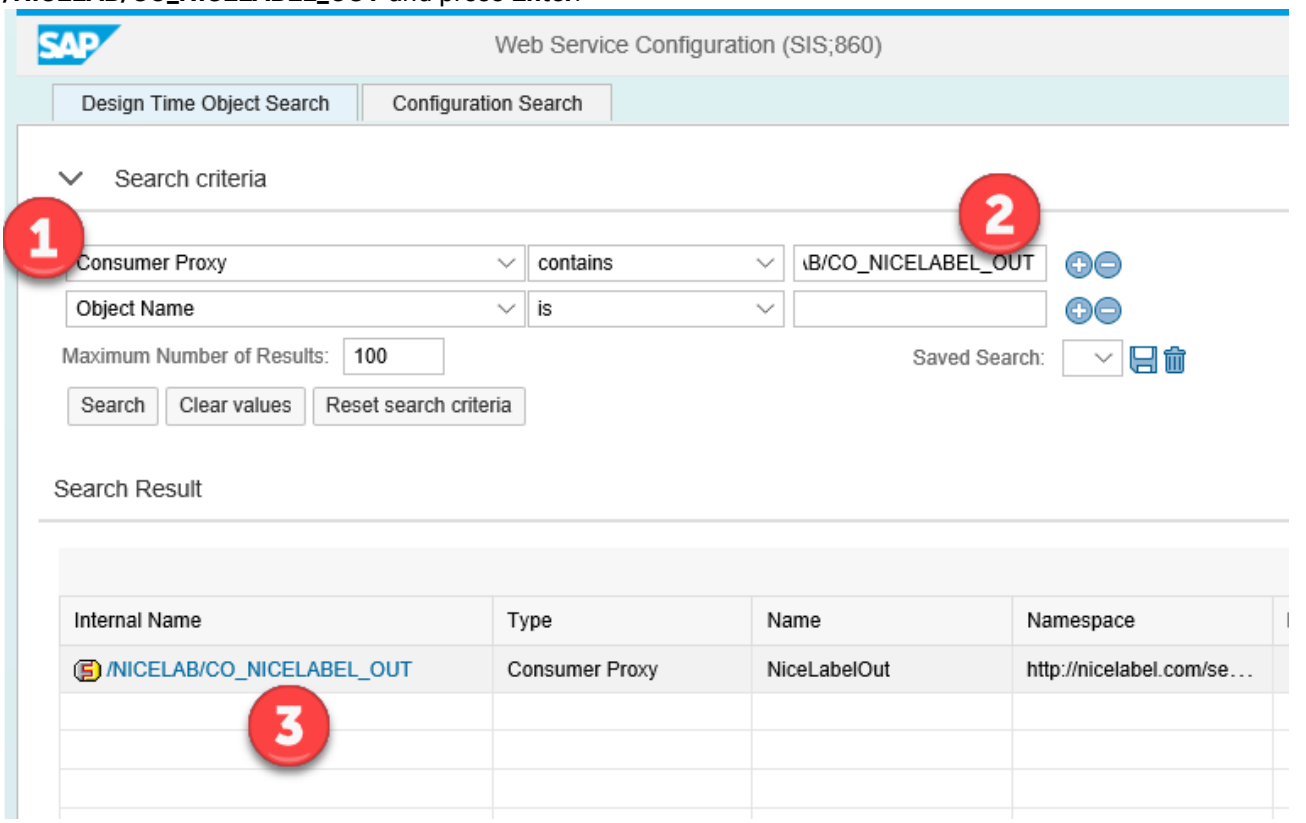
Configuring logical port

To configure the logical port, do the following:

1. In **SAP GUI**, run the transaction **SOAMANAGER**.
   The **SOA Management** opens in the browser.
2. In the **Service Administration** tab, click **Web Service Configuration**.

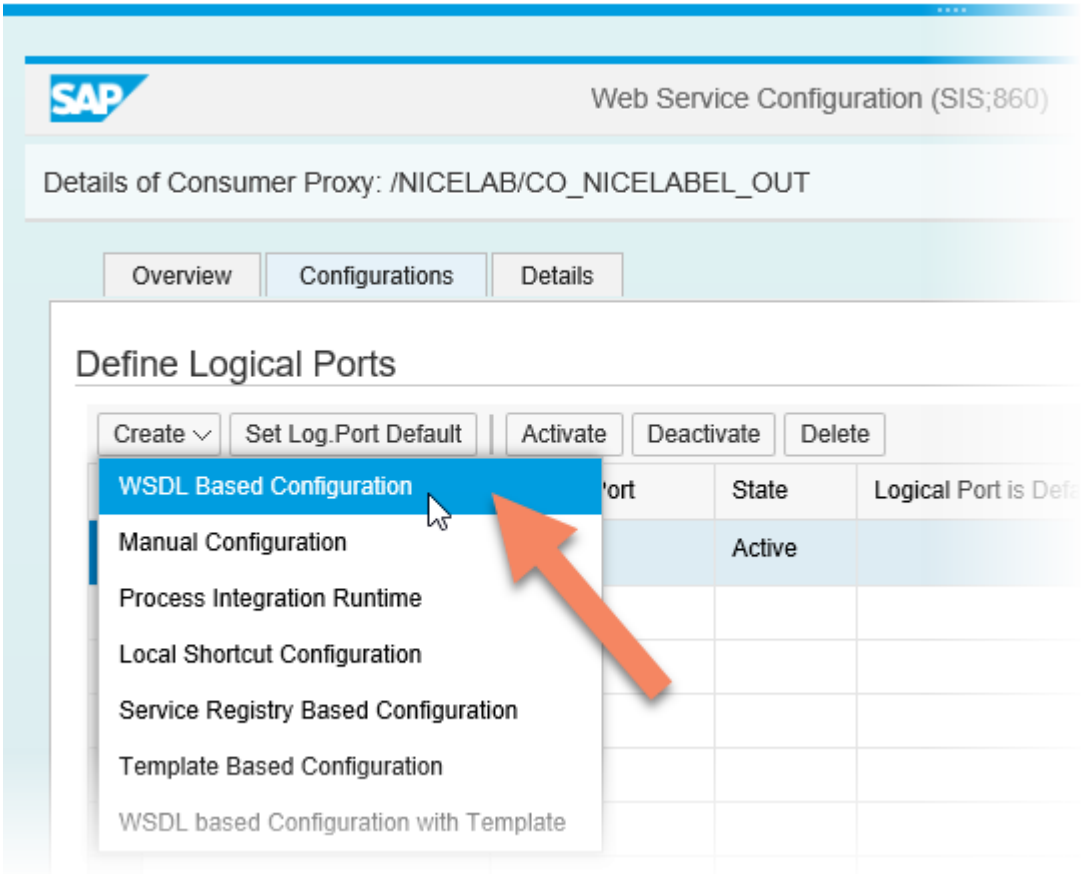


3. In the **Search criteria** drop-down box, select **Consumer Proxy**.
4. In **Search Pattern,** enter the name of the consumer proxy as provided by the ABAP Package. Enter **/NICELAB/CO_NICELABEL_OUT** and press **Enter**.



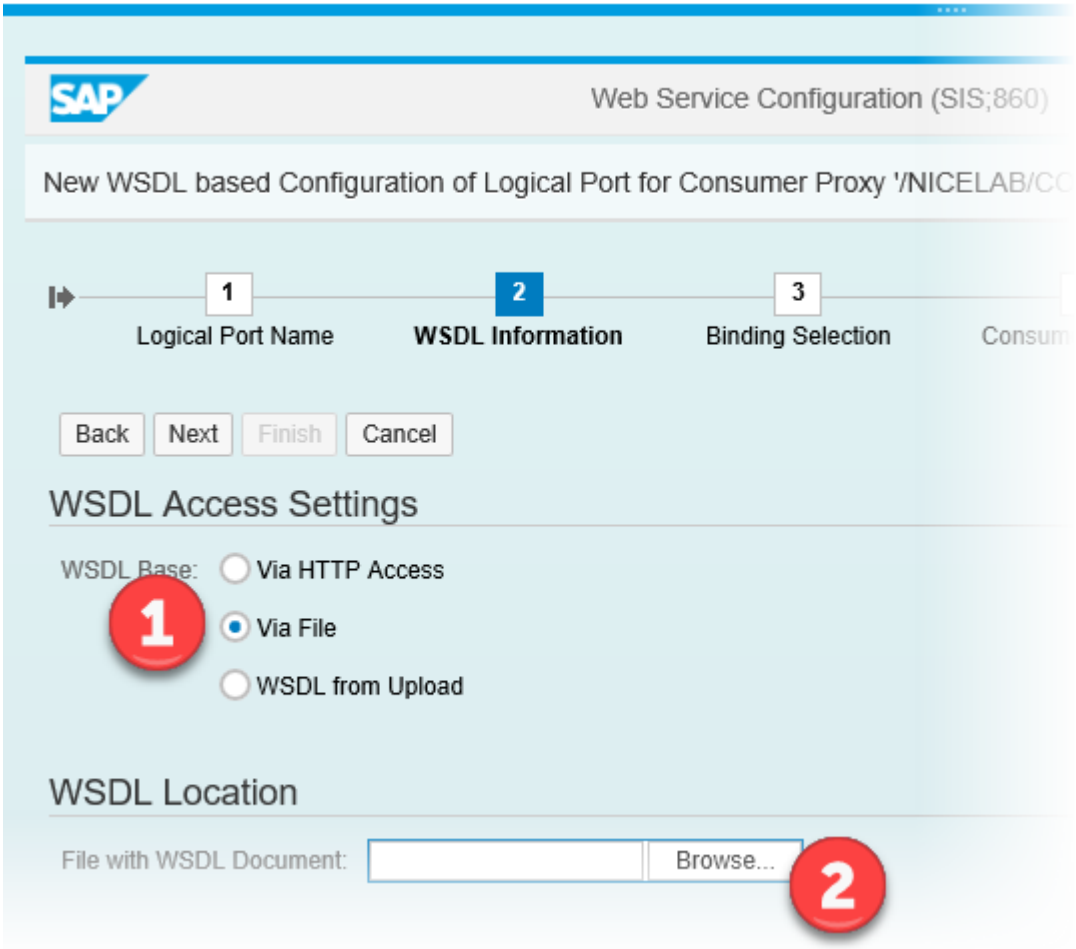
5. Click the **/NICELAB/CO_NICELABEL_OUT** entry in the search results.

6. In the **Configurations** tab, click **Create**.
   In the drop-down, select **WSDL Based Configuration**.



7. In the **Logical Port Name** step, enter the name that identifies this logical port.
8. In the **Description** field, enter the text that describes the logical port.
9. Click **Next**.
10. In the **WSDL Information** step, select **Via File** and click **Browse**. Then browse for the WSDL file you downloaded from the BTP cockpit. You can also use the WSDL file provided in the ABAP Package

bundle (see folder \Integrations\CPI Configuration).



11. Click **Next**.
12. In the **Binding Selection** step, click **Next**.
13. In the **Consumer Security** step, type in the user name and password in the respective fields. This is the user you configured in the **SAP BTP Cockpit** for the role collection **NiceLabel_MessageSend** in the previous steps.
14. Click **Next**.
15. In the **HTTP Settings** step, make sure the URL Access Path is set to **/cxf/NiceLabelService**.
16. Make sure the **Computer Name of Access URL** is set to the endpoint from your SAP Integration Suite. Do not include the leading “https://” text.
    a. If you have used the WSDL downloaded from your SAP Integration Suite, the endpoint should already be correct.
    b. If you have used the WSDL provided with the ABAP Package bundle, you will have to change the URL to use endpoint address from your SAP Integration Suite.
17. Click **Next**.
18. In the **SOAP Protocol** step, click **Next**.
19. In the **Identifiable Business Context** step, click **Next**.
20. In the **Operation Settings** step, click **Next**.
21. Click **Finish**.

Configuring Cloud Trigger access

You must enable the access to the Cloud Trigger. You must obtain the subscription key that allows you to consume the Cloud Trigger API.

For more information, see the chapter:

- **Generating a subscription key to consume Loftware Cloud APIs**

Using ABAP Package features

Configuring the demo transaction

The ABAP Package includes a demo transaction developed in ABAP that demonstrates the practical use of the API. The demo transaction provides a user interface for testing the underlying API. The demo transaction collects the provided data and sends it to ABAP Package. ABAP Package encapsulates data in XML message and sends it to Loftware Automation for processing.

Before you can run the demo transaction, you must configure it. The demo transaction reads the communication method (SOA or RFC) and the destination name (logical port or HTTP destination) from the configuration table.

You must configure the following:

1. Run tcode **/n/NICELAB/IF_CTRL** to open the configuration table.
2. If you intend to use Web Service (SOA) connection to Loftware Automation, do the following:
    a. Add Config ID **OPERATION_MODE**.
    For value, type **SOA**.
    b. Add Config ID **SOA_LP**.
    For value, type the name of the logical port you defined for Web Service connection in tcode **SOAMANAGER**.
3. If you intend to use RFC (HTTP REST) connection to Loftware Automation, do the following:
    a. Add Config ID **OPERATION_MODE**.
    For value, type **HTTP**.
    b. Add Config ID **HTTP_RFC**.
    For value, type the name of the HTTP destination you defined in tcode **SM59**.
4. Save changes.

For details see **Configuring system-wide defaults** chapter on page **30**.

Using the demo transaction

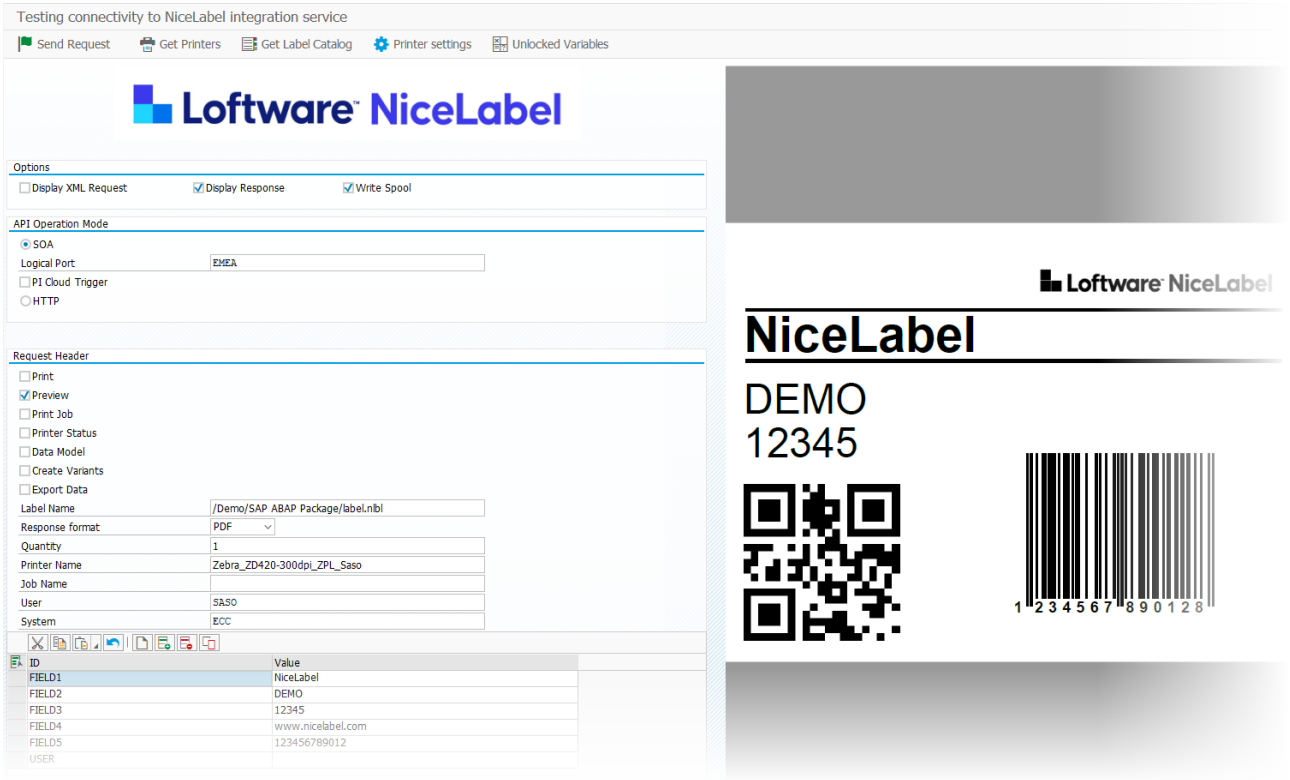
The demo transaction receives and displays feedback from Loftware Automation. Possible feedback is the success of the printing process, printer live status, label preview, list of available labels in the DMS, list of available printers, list of variables from the selected label, binary print job, etc.

The responses for label catalog or printer names calls are cached. To force a call for label catalog or printer names, use the appropriate button in the demo transaction (Get Printer, Get Label Catalog).

NOTE: The prerequisite for running the demo transaction is an already deployed configuration in Loftware Automation Manager.

To run the demo transaction, do the following:

1. In **SAP GUI**, run the transaction **/n/NICELAB/IF_DEMO**.
   The **Testing connectivity to Loftware integration service** transaction appears.



2. Enable **Display Request** if you want to display the generated XML payload on the screen instead of sending it to Loftware Automation. You would use this option to analyze the data in the composed XML message.
3. Enable **Display Response** to display the status feedback from Loftware Automation. If any errors occur during processing, Loftware Automation returns a detailed error message, and the demo program displays it on-screen. Make sure this option is always enabled.
4. Enable **Write Spool** to log all messages to/from Loftware Automation in SAP Spooler (tcode SP01). Each request to Loftware Automation and all responses from Loftware Automation are logged in SAP Spooler.
5. In **API Operation mode**, select the transport method you want to use for communication with Loftware Automation.
   a. **SOA.** Select this option if you want to send the data into Web Service trigger in Loftware Automation. Click **Browse** and select the logical port you have defined earlier in the topic **Configuring SOA (Web Service) destination** on page 21.

      This operation mode is also used when sending print requests through SAP PI/PO or SAP CPI infrastructures.

   b. **HTTP.** Select this option if you want to send the data to the HTTP trigger in Loftware Automation. Click **Browse** and select the RFC Destination you have defined earlier in topic **Configuring HTTP (RFC) destination** on page 25.
6. In the **Request Header** section, the following parameters are configurable.
   a. **Print.** Enable this option to print labels.
   b. **Preview.** Enable this option to generate and display label previews in SAP GUI.
   c. **Print job.** Enable this option to generate and save print jobs in SAP Spooler. Make sure you enable the **Write to spool** option. Print jobs contain the printing instructions for the selected printer models. These instructions are printer commands for the target printer (Zebra Programming Language “ZPL” for Zebra printers, SATO Programming Language “SPL” for

SATO printers, etc.) You can send print jobs from SAP Spooler to the printer (passthrough mode).

NOTE: It is best to use Loftware printer drivers to create print jobs.

d. **Printer Status.** Enable this option to display information about the live printer status.

You can combine the flags Print, Preview, and Printer Status within the same request. Loftware Automation executes all received methods.

e. **Label Name.** Specifies the file name of the label template to print. Make sure the label name contains the extension (.NLBL) as well. You can click the **Search Help** button  next to the FORMAT edit field to display the label catalog (list of all labels in your DMS). Click the **Get Label Catalog** button to retrieve the list of labels first.

NOTE: For Loftware Cloud products, the label catalog is created synchronously. For on-premise NiceLabel LMS, the label catalog is created asynchronously in the background. It might take a while to become available (several minutes). This depends on the number of labels. Until the label catalog is ready, you receive an empty list.

For a quick test, you can use the included sample label file, **`label.nlbl`**. If copied the label template in the same folder with your Automation configuration (.MISX file), you can reference the label by its filename. If you have copied the label template to some other folder, include a full path to the file.

For more information about using the file names, see chapter **Providing a label** name on page 67.

f. **Response format.** Specifies the format of the label preview. You can choose between PDF, PNG, and JPEG.
g. **Quantity.** Specifies the number of label copies to print.
h. **Printer Name.** Specifies the name of Windows printer driver on the server, on which Loftware Automation runs. You can click the **Search Help** button  next to the PRINTER_NAME field to display the list of printers. The request for the printer list goes to Automation.

NOTE: This is not the SAP printer name. This is the Windows name of the printer driver installed on a computer with Loftware Automation.

i. **Job Name.** Specifies the name of the print job as displayed in the Windows Spooler. This is an optional parameter. By default, the job name is the same as the label name. Use this option to rename the print job.
j. **User.** Specifies the user name that requests the printing. This is an optional parameter and is currently not in use in the Automation configuration. You might use it to provide the information to the label.
k. **System.** Specifies the SAP system information. This is an optional parameter and is currently not in use in the Automation configuration. You might use it to provide the information to the label.
l. **Tray.** Specified the printer's paper tray name from where the media must be taken. Make sure to use the tray name as provided in the printer driver.
m. **Printer settings.** The printer settings define the printer configuration for the specific media, including settings like speed, darkness, sensor type, etc. Loftware gets the printer settings stored in the label template, recalls them from the printer driver, but this option allows you to apply your own printer settings.
n. **Label settings**. The label settings define the layout of the label template, including label dimensions, orientation, number of labels on the page, margins to the media edge, offsets

between labels etc. These settings are usually defined by the label designer in a NLBL file. However, you can override the settings at print time. This allows you to maintain less label templates and reuse the same label template for different printer and media types. For example, you can define a label template to print on a roll of labels, but for another print scenario you can print the same label on an A4/Letter page of labels multiple time and in different rotation.

7. The table at the bottom of the transaction displays the list of variables defined in the selected label. Each time you select a new label, the values for the variables are taken from the label. You can accept the suggested default values for variables or enter your own. You can also add your variables (rows). Loftware Automation does not impose any limitation on the number of name-value pairs you want to use.
8. When ready, click **Send Request** (or press F8).

Requesting a list of printers

To request a list of printers available for label printing, ABAP Package must set the **<PRINTERS />** XML element to **True**. The returned list contains printer names that Loftware Automation can use for label printing. These can be local printers with drivers installed on Automation's computer, or cloud-connected printers.

When you request a list of printers, the ABAP Package stores it in the table /NICELAB/IF_PRNT. The table contains the combination of printer name and identifier of the Automation server, so ABAP Package knows which printers are available from each Automation server. The "identifier" is a combination of two fields: (1) a type of the Automation endpoint (e.g., "Web Service", "HTTP Request") and name of the endpoint (e.g., name of the Logical Port or RFC destination). Each printer in the list is also identified as the local printer (installed on the same computer with Automation) or the cloud-connected printer.

The printer name is encoded in the **<PRINTERNAME />** element inside the XML header. When you select a particular printer, ABAP Package lets Loftware Automation know if it is a local or cloud printer, so Loftware Automation will process the request correctly. For cloud printers, the additional flag **<CLOUD_PRINT />** in the header is set to True.

Two types of printers can be requested:

- **Only local printers.** The list of printers is acquired from the computer that runs Loftware Automation. Loftware Automation returns the list of visible printers. Printer accessibility is determined by the user permissions of the Windows user account that you use to run the Automation Service process.

  In this case, the <Header> element must include:
  - **<PRINTERS />** element set to **True**.

- **Local and Cloud printers.** With your Loftware Cloud subscription, you also get access to using cloud-connected printers. In this case, the printer is registered with the Loftware Cloud and accepts print jobs directly from the Loftware Cloud (not from Automation). You do not have to install the printer drivers for cloud-connected printers.

  NOTE: This functionality requires Loftware Cloud Business or above.

  The <Header> element must include:
  - **<PRINTERS />** element set to **True**.

- You must configure the OCP-APIM-SUBSCRIPTION-KEY and API-VERSION keys in the **/NICELAB/IF_CTRL** configuration table. For details see **Configuring system-wide defaults** chapter on page **30**.
  These two configurable items will be encoded as **<CLOUD_APIKEY />** and **<CLOUD_APIVERSION />** elements in the <Header /> element.

For XML structure of the response, see chapter **Structure of <PrinterList />** on page **97**.

NOTE: A response will contain a list of all local printers that are installed on a computer with Automation. However, all these printers might not be accessible by Automation, dependent on how you configured printer access in Control Center.

Printing a label or report

To print a label or report to local printers, the <Header /> element must contain the following:

- <PRINT /> element set to **True**.
- <FORMAT /> element to set the label or report template created with Loftware Designer.
- <PRINTERNAME /> element to set the printer. This is an optional element. When not provided, the label prints to the printer that is saved with the label template.
- <QUANTITY /> element to provide the number of labels to print.
- <QUANTITY_LABELCOPIES /> element to provide the number of copies to print per each label. This is an optional element. When not provided, each label prints in one copy.
- Optionally. <TRAY /> element to provide name of the paper tray (paper bin) which contains the label media for printing.

The label and printer determination must be done within SAP. Automation requires the exact label template name as available from the Document Management System and exact Windows printer name as available on the computer, where Automation is installed, or exact cloud printer name. If required, the label and printer determination can be performed in Loftware Automation as well but is subject to additional configuration.

Automation automatically identifies whether the data in XML is provided for label or report printing. You must make sure to provide the appropriate label or report template in the <FORMAT /> element. Names of SAP data sources must match the names of data sources defined in a label or report template to ensure proper value mapping. For example, when you have a field MATNR in an SAP transaction, the label template must contain a variable of the same name.

You can use Automation to generate data model files to help you create data sources in the label or report templates. For details, see **Generating a field catalog** chapter on page **69**.

For XML structure of the response, see chapter **Structure of <PrinterList />** on page **97**.

NOTE: You must insert at least one <Data> element in your XML payload, even if you provide no key-value pairs to the label. In case of empty element, use the syntax <Data></Data> (not the shorten variant <Data />).

Printing labels to cloud printers

NOTE: This functionality requires Loftware Cloud Business or above.

ABAP Package can print labels to cloud-connected printers. These are printers that reside on-premise, but have internet connectivity and are registered with your Loftware Cloud subscription. When you switch them on, they establish a connection with Loftware Cloud and keep it open. Loftware Cloud API can deliver print

jobs (e.g., ZPL content for Zebra printers) directly to a printer. You do not need any on-prem printer driver installed.

Loftware Automation is pre-configured to receive the information for your cloud printers from ABAP Package and can send label print request to the Loftware Cloud API, which in turn prints the label.

To print labels to a cloud printer, you have to do the following:

- You must configure the OCP-APIM-SUBSCRIPTION-KEY and API-VERSION keys in the **/NICELAB/IF_CTRL** configuration table. For details see **Configuring system-wide defaults** chapter on page **30**.
- Request a list of available printers from Automation. Because of step (1) above, the Automation will combine available local and cloud printers in the response. ABAP Package saves the received printer list in a table /NICELAB/IF_PRNT. Each printer includes a Boolean field identifying it as a local or cloud printer.
- To print a label to a cloud printer, use the cloud printer name in the <PRINTERNAME> element when issuing a print request. When you select cloud printer, ABAP Package will include the field <CLOUD_PRINT> with value **True** in the <Header /> element, which notifies Automation to use Cloud Print API. You also have to set the main printing flag <PRINT> to **True**.
- Optional. You can specify the label template version with **<CLOUD_LABELVERSION />** element. If omitted, the last published version is used.

Previewing a label or report

To preview a label or report, the <Header /> element must contain the following:

- <PREVIEW /> element set to **True**.
- <PREVIEW_FORMAT /> set to **"PDF"**, **"PNG"** or **"JPG"**.
  This element is optional. Without explicitly defining a return value, you will get a preview formatted as PDF.

The preview is returned as base64-encoded data in the XML response.
PDF can be a multi-page document, while bitmap preview returns just the first page in a batch.

For XML structure of the response, see chapter **Structure of <PrinterList />** on page **97**.

NOTE: You must insert at least one <Data> element in your XML payload, even if you provide no key-value pairs to the label. In case of empty element, you must use the syntax <Data></Data> (not the shorten variant <Data />).

Emailing label preview

To email label preview as attachment to the list of recipients, the <Header /> element must contain the following:

- <EMAIL_RECIPIENTS> element with the comma-separated list of email recipients

The generated label preview will be sent to all recipients in the list.

Before you the emails can be sent, you must configure the email-related properties in the Automation configuration for your selected trigger type.

Do the following:

1. Open Automation configuration (MISX file) for editing in the **Loftware Automation Builder**.
2. Open properties of the trigger type you intend to use from SAP. This is dependent on your preferred communication type from SAP to Loftware (e. g. RFC or Web Service call).
3. Go to **Variables** tab.
4. Define default value for the variable **_EMAIL_FROM**. This should be the sender of the emails from your server. For example noreply@yourdomain.com.
5. Define default value for the variable **_EMAIL_SMTPSERVER**. This should be the IP address or FQDN (fully qualified domain name) of your SMTP server. Automation will send emails through this server.
    a. OPTIONALLY. Set default value for the variables **_EMAIL_SMTPSERVER_USERNAME** and **_EMAIL_SMTPSERVER_PASSWORD** if your SMTP server demands authentication.
6. OPTIONALLY. Set default value for the variable _EMAIL_SUBJECT. This contains the value for **Subject** in the email.
7. OPTIONALLY. Set default value for the variable _EMAIL_MESSAGE. This contains the value for **body** in the email. It must have the HTML formatting.
8. Save the Automation configuration.
9. Deploy the new Automation configuration.

NOTE: The email is sent only when the **EMAIL_RECIPIENTS** element is not empty.

Adjusting label settings at print-time

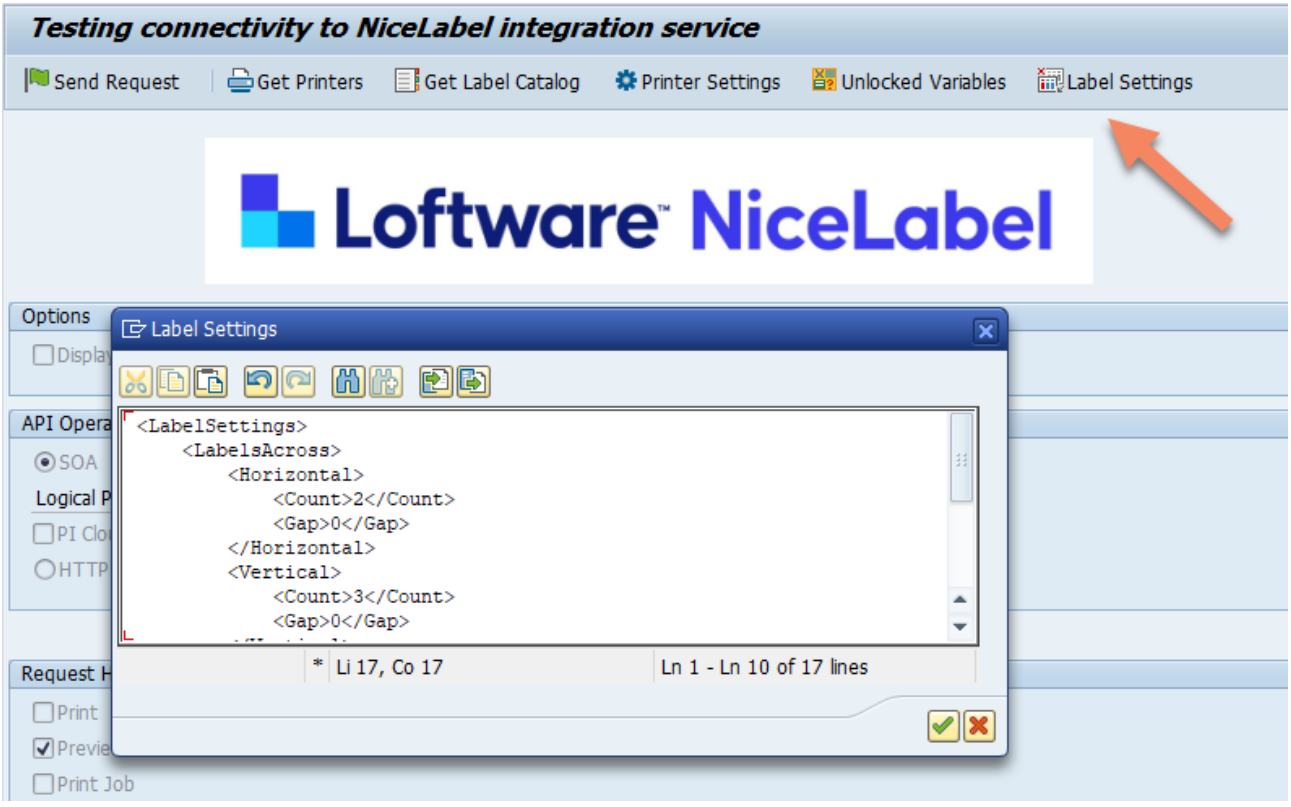
The label settings are width and height, number of labels in horizontal or vertical dimension, gaps between labels, margins from the media edge, orientation, rotation and similar. You define these settings while you design your label template in Loftware Designer and they become part of the template's definition.

Loftware Automation allows for dynamic label template changes at print time. You can modify some of the label settings at print-time – they are applied to the print job, but not saved in the label template. This helps to minimize the number of label templates you must maintain. For example, you can create a label template designed for label printers in dimensions 105×148 mm (which is roughly 4×6") and print the same label template on different printers with different orientation and number of templates in vertical and horizontal dimension.

Printing on a label printer (such as Zebra).	Printing same label template with overridden label settings to print it on the laser printer using A5 page size. Two label templates fit on the A5 page.	Printing same label template with overridden label settings to print it on the laser printer using A4 page size. Four label templates fit on the A4 page.

To adjust the label settings at the print time, you must provide the label-settings modifier XML payload in the Header element **LABELSETTINGS**. The accepted value is the XML payload described for the Label setting parameter in the Automation's action **Set Print Parameter**.

To test this functionality in the demo transaction, click the button **Label Settings**.



For the details, see Loftware online help.

Checking printer live status

To check the status of the specified printer, the <Header /> element must contain the following:

- <PRINTERSTATUS /> element set to **True**.

Furthermore, when you select the status of a cloud-connected printer, ABAP Package includes these additional fields in the <Header /> element:

- <CLOUD_PRINT /> with value **True**.
- <CLOUD_APIKEY> with the value of OCP-APIM-SUBSCRIPTION-KEY configurable item from **/NICELAB/IF_CTRL** table.
- <CLOUD_APIVERSION> with the value of API-VERSION configurable item from **/NICELAB/IF_CTRL** table.

Automation will check the printer status and return it in a synchronous response.

For details of exchanged data structures, see **Data exchange** on page **91**.

Overrides

Overriding print settings (label template, printer name, quantity)

A print request from ABAP Package to Loftware Automation usually contains data for one label or report. The XML payload contains one <Header /> element that provides label or report template name, printer name, and quantity of labels. The <Data /> segment contains data to print on a label or report.

You can provide several <Data /> elements, each providing data for a new label or report, but using the same label template and printer name as provided in the <Header /> element.

You can override the print settings for each <Data /> element if you want to print data using a different label template on a different printer, but all within one request from ABAP Package to Loftware Automation.
You must provide the data for label template name, printer name, or quantity inside each <Data /> element like this:

- For label template name use variable _FORMAT
- For printer name use variable _PRINTERNAME
- For quantity use variable _QUANTITY

NOTE: When you decide to use data override in <Data /> elements, you must use the override in all <Data /> elements.

```
<?xml version="1.0"?>
<LABELS>
    <Header>
        <FORMAT>label.nlbl</FORMAT>
        <QUANTITY>1</QUANTITY>
        <PRINTERNAME>ZEBRA R-402</PRINTERNAME>
        <PRINT>True</PRINT>
    </Header>
    <Data>
        <Item Id="_FORMAT">label1.nlbl</Item>
```

```
        <Item Id="FIELD1">Loftware</Item>
        <Item Id="FIELD2">DEMO</Item>
        <Item Id="FIELD3">12345</Item>
        <Item Id="FIELD4">www.loftware.com</Item>
        <Item Id="FIELD5">123456789012</Item>
    </Data>
    <Data>
        <Item Id="_FORMAT">label2.nlbl</Item>
        <Item Id="FIELD1">Loftware</Item>
        <Item Id="FIELD2">DEMO</Item>
        <Item Id="FIELD3">12345</Item>
        <Item Id="FIELD4">www.loftware.com</Item>
        <Item Id="FIELD5">123456789012</Item>
    </Data>
</LABELS>
```

In the example above, the XML payload provides two <Data /> segments. Data from the first segment will print on label template `label1.nlbl`. Data from the second segment will print on label template `label2.nlbl`.

You can use similar overrides for all fields available within the <Header /> element by prefixing underscore to the field name, e.g., **FORMAT** -> **_FORMAT**.

Overriding printer settings

To override the printer settings, the <Header /> element must contain the following:

- <PRINTERSETTINGS /> element containing base64-encoded DEVMODE for the printer specified in <PRINTERNAME /> element.

The “printer settings” are settings in the printer driver that define details for the printout, such as to use thermal printing mode or direct printing mode, details for the printer sensors, offsets, and similar.

When designing a label template in Loftware software, you can store printer settings in various places (e.g., hardcode it in the label templates itself, recall it from the printer driver, rely on the settings as configured in the printer, etc.)

When you do not specify <PRINTERSETTINGS /> value, Loftware Automation will use the printer setting as configured with the label template / selected printer. When you include <PRINTERSETTINGS /> the current printer settings will be overridden with provided settings. This is a useful option to fine-tune printout for cloud-connected printers, where there are no printer drivers to configure, but also works for local printers.

The DEVMODE is the entire printer driver structure. To extract the settings and base64-encode them, you can use Loftware Designer to create such an application. You would use the built-in action **Set Print Parameter**.

Overriding email settings

The list of email recipients to receive the label preview is configured with an element EMAIL_RECIPIENTS in the Header element in the XML payload. You can provide a different list of recipients with each call to the ABAP Package.

The other email-related settings, such as the email server properties, email subject, email message and subject field, are hardcoded in the Automation configuration. When needed, you can override these fields

with each call to the ABAP Package as well. There is no definition for them available in the Header structure, but you can provide them in each Data element as you do with the rest of key-value pairs that you want to print in the label.

You must provide the data for email settings inside each <Data /> element like this:

- For the SMTP server use variable **_EMAIL_SMTPSERVER**
- For the email server SMTP credentials use the variables **_EMAIL_SMTPSERVER_USERNAME** and **_EMAIL_SMTPSERVER_PASSWORD**
- For the from field (email of the sender) use the variable **_EMAIL_FROM**
- For the subject field use the variable **_EMAIL_SUBJECT**
- For the email message use the variable **_EMAIL_MESSAGE**

Logging data to SAP Spooler (SP01)

You can configure the ABAP Package to save all events in SAP Spooler (tcode SP01).

This functionality allows for complete tracking of Loftware execution inside the SAP system. Each outbound request sent to Loftware Automation and each inbound response coming from Loftware Automation are stored as items with a unique Spool number.

The function that updates the event data in SP01 is available as **/NICELAB/IF_RW_SPOOLJOB_PARAMS** and is developed from the SAP standard function **RSPO_RW_SPOOLJOB_PARAMS**.

In the example below, there is a log of two requests sent to Loftware Automation. The first one at 13:05 and the second at 13:19. You can see all data that was sent out. The item with Spool id **532065** contains the outbound message, displayed as parsed XML at the beginning and the raw data below.

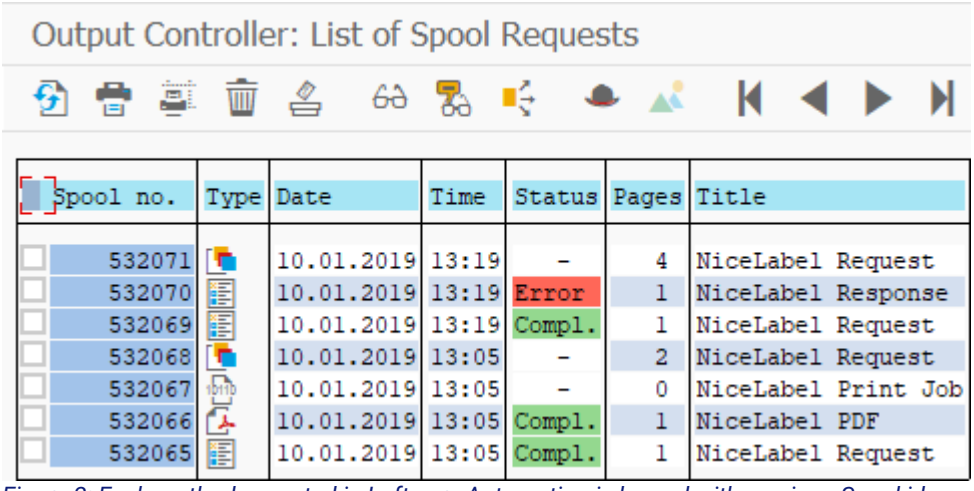


Figure 2: Each method executed in Loftware Automation is logged with a unique Spool id

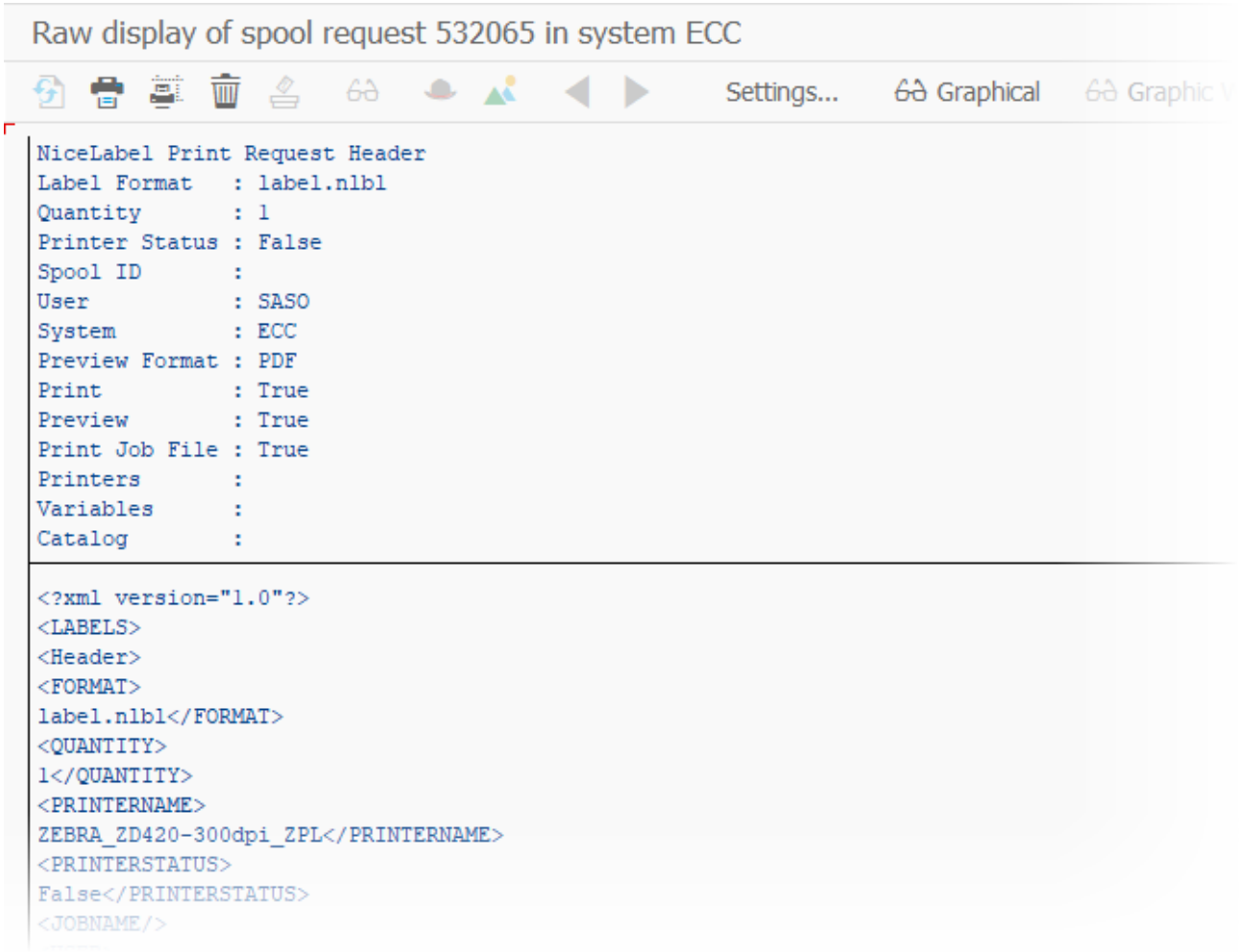


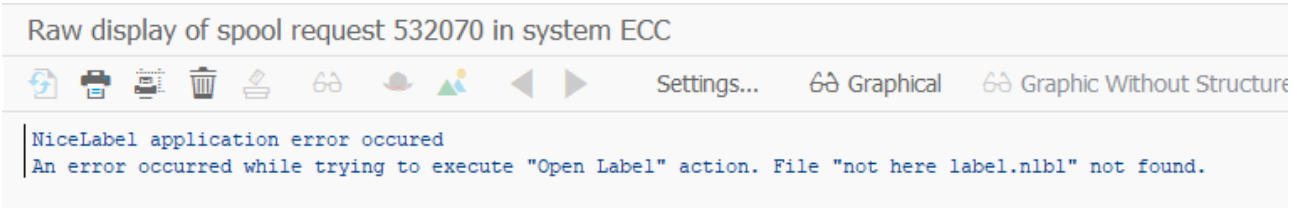
Figure 3: Details for the outbound message to Loftware Automation

Each log also contains the endpoint type and name for you to see to which destination (Loftware Automation server) the request has been sent. For example, endpoint type **HTTP Request** and endpoint name **EMEA** identify the request as RFC call to RFC destination configured as “EMEA” (in tcode SM59).

This request executed three methods in Loftware Automation:

1. **To print a label (532065).** Three was no error reported from Loftware Automation. Printing completed successfully.
2. **To receive a label preview as PDF (532066).** The label preview is stored as a PDF document. The preview is available to meet the regulatory requirements, or you can use it as proof of printing.
3. **To receive a binary print job (532067).** Loftware Automation has returned a print job. The print job contains the label with merged SAP data. The data has been converted into the programming language of the selected printer. You can send this data to the printer defined in SAP and the label will print. You would use this function if you do not want Loftware Automation to send data to your printer directly, and if you want to have printing control inside SAP.

The second request (from 13:19) failed. The outbound request was sent to Loftware Automation, but there was an error while processing the data. Details in Loftware Response (Spool id **532070**) reveal that the label name was not found in your DMS. Whenever there is a data processing issue in Loftware Automation, the details about the error return to SAP in a synchronous response.



You can execute more than just one method using a single request to Loftware Automation. For example, you can execute label print and label preview methods using a single message sent to Loftware Automation. “Label print” response logs under a unique Spool id. “Label preview” response also logs logged under a unique Spool id. To keep the things organized and to ensure optimal user experience, all Spool items from the same response are grouped into a **composite document**. This is a new Spool id that contains the individual Spool ids. The composite document appears only when your request enables two or more methods.

NOTE: You cannot combine the PRINT_JOB request with any other request. It must be used in the unique request.

In this example, the Spool id **532068** is a composite document containing all subitems that were returned in the response from Loftware Automation.

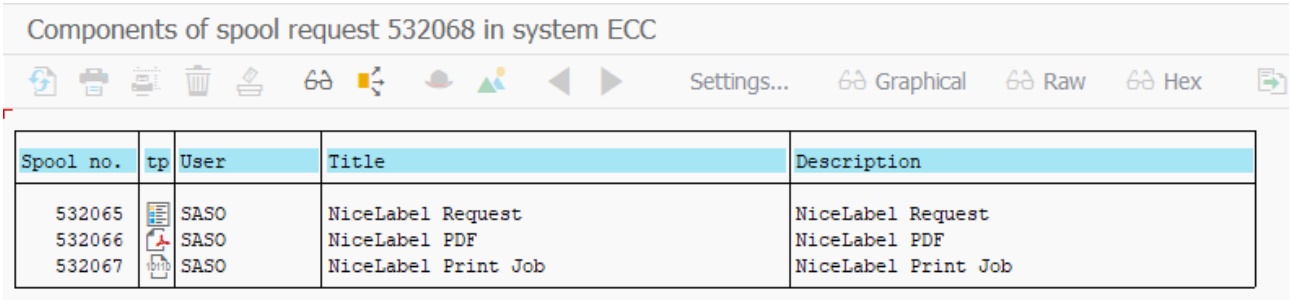


Figure 4: All subitems from a single request to Loftware Automation are grouped together

To enable writing events to SAP Spooler, do any of the following:

- In demo transaction, enable the option **Write Spool**
- To enable this option as a default, add Config ID **WRITE_SPOOL** to the configuration table and set its value to **True**. For more information, see chapter **Configuring system-wide defaults** on page 30.
- Enable or disable the writing to Spool option with each request to the ABAP Package.

The items in SAP Spooler are associated with the specific output device:

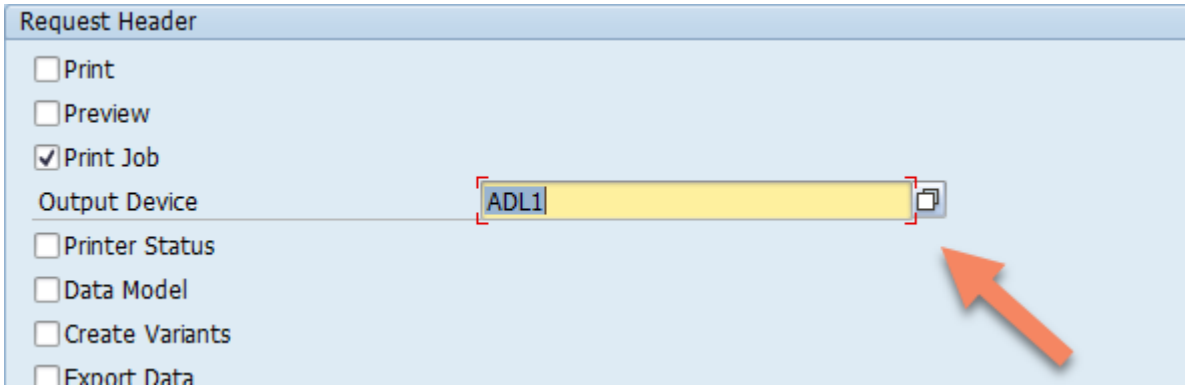
- If you request PRINT_JOB, the items are associated with the **NiceLabel Raw Printer (NLBN)** output device. You can override this default output device with your own when making a call to the ABAP Package.
- In all other cases, the items are associated with the **NiceLabel Request to Spool (NLRS)** output device.

Defining your own Output Device for print jobs in SAP Spooler

Every print job in SAP Spooler is associated with some output device. When you use ABAP Package to write print job in the SAP Spooler, the built-in output device **NiceLabel Raw Printer** is used for the associated output device. When you want to send print job to the real output device, you must change the output device at print time.

If you know the name of your output device in advance, you can command the ABAP Package to associate item in the SAP Spooler with that output device, so you don't have to modify it at print time. The output device must already exist in your SAP system.

For an example usage, see the provided demo transaction. When you select the option **Print Job**, you can select one of the already configured **Output Devices**. Make sure, you also enable the **Write Spool** check box in the **Options** panel to write data to SAP Spooler.



See source code the demo transaction to understand how to programmable execute a call to ABAP Package and provide the output device name.

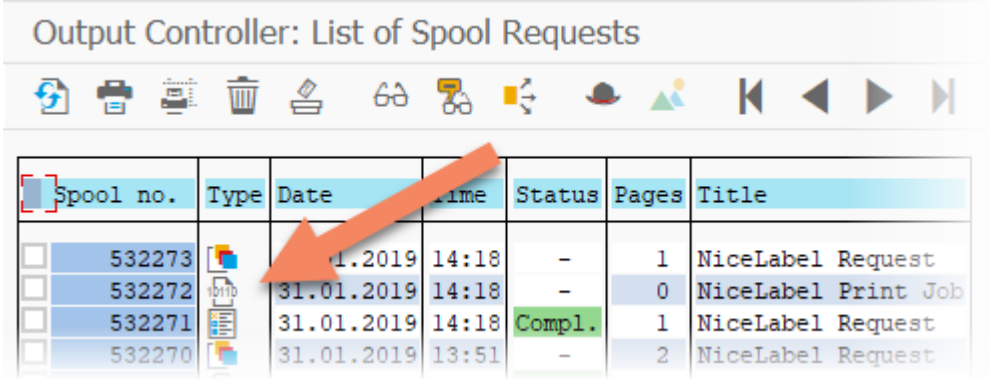
Printing binary print jobs from SAP

When the ABAP Package sends value **True** in the **<PRINT_JOB />** XML element, Loftware Automation returns the binary print job. ABAP Package then saves the print job inside SAP Spooler. By default, the job is associated with the **NiceLabel Raw Printer (NLBL)** output device (which is provided with the ABAP Package).

NOTE: You can override the NiceLabel Raw Printer output device and use your own output device. See the chapter **Defining your own Output Device for print jobs in SAP Spooler** on page 65.

To send the binary print job to the printer:

1. Run tcode **SP01**.
2. Find the print event you have executed through the ABAP Package.

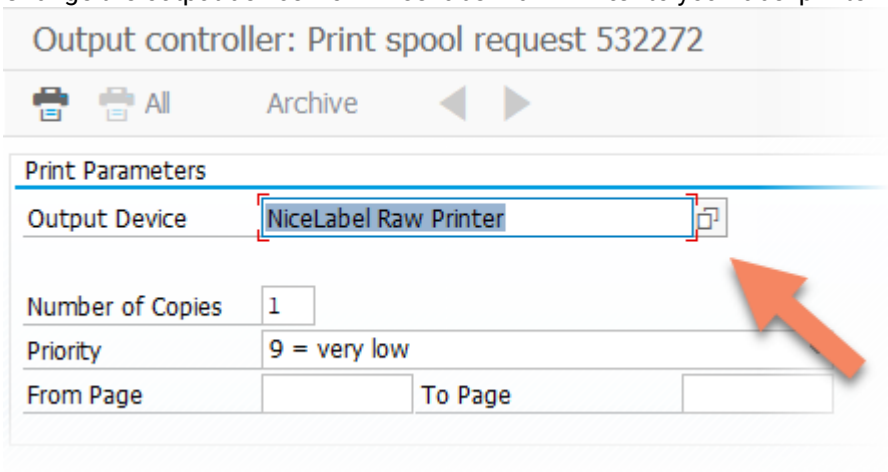


3. Click the icon representing the binary data.

   The spool request displays as a graphical preview. As the request contains binary characters, they do not display properly on-screen.

4. Click the **Print with changed parameters…** button.

5. Change the output device from **NiceLabel Raw Printer** to your label printer.



Prerequisites:

1. You must have the output device already created in tcode SPAD.
2. The selected output device must be compatible with the print job format you requested from Loftware Automation. The printer model associated with the SAP output device must understand the printer programming language provided in the binary print job from Loftware Automation.

Providing a label name

When providing a label name, use any of the following syntaxes:

- **Label name without the path.** Loftware Automation looks for the label file in the same folder, where the configuration file (.MISX) is located and folders `..\Labels` and `.\Labels`.

  ```
  label.nlbl
  ```

- **Label name with the relative path.** The root folder is the folder with the Automation configuration file (.MISX).

  ```
  ..\DistributionLabels\Gen1\label.nlbl
  ```

- **A full path to the label in DMS, including the server name.** This syntax works for on-premise and Loftware Cloud products.

  ```
  https://myaccount.onnicelabel.com:8080/DistributionLabels/Gen1/label.nlbl
  ```

- **A full path to the label in your DMS w/o the server name.** This syntax works for on-premise and Loftware Cloud products. The name must begin with the slash character (/), and the path must be provided from the Control Center's Documents root folder.

```
/DistributionLabels/Gen1/label.nlbl
```

Creating a label catalog

To request a label catalog, ABAP Package must set the **<CATALOG />** XML element to **True**.

Optionally, you can specify a starting folder for your label templates in the **<CATALOG_ROOT />** XML element. When no value is provided, Automation will create the catalog from the root folder in the DMS.

The demo transaction caches the received label catalog in the internal table.

The process to generate the label catalog in Automation is different depending on the Loftware product you have.

For the response XML structure, see chapter **Structure of <LabelCatalog />** on page **99**.

Requesting label catalog for Loftware Cloud

Automation uses native Document API to create a label catalog when you use Loftware Cloud products. This approach provides the fastest possible method to generate the catalog. ABAP Package retrieves a label catalog in a response to a synchronous call (ABAP Package waits for Automation to generate the label catalog on the fly).

Automation must receive a subscription key from the ABAP Package to successfully consume the Document API. The subscription key is the authentication method to access the Document API. The demo transaction will read the subscription key from the configuration table and include it inside the **Header** element of the XML payload.

NOTE: The user account that you used to activate Loftware Automation must belong in the access role with the permissions to read folders and label templates to generate the label catalog.

You must do the following:

1. Generate the subscription key for your Loftware Cloud account.
2. Add items to the ABAP Package's configuration table **/NICELAB/IF_CTRL**.

For more information, see the chapter **Generating a subscription key to consume Loftware Cloud APIs** in this document.

Requesting label catalog for NiceLabel LMS

There is no Document API available for NiceLabel LMS products. Automation generates the label catalog with a help of a command-line utility. ABAP Package retrieves the current version of a label catalog, while Automation starts generating a new label catalog in a background thread (asynchronously). The generation of the label catalog can take several minutes, dependent on the number of label templates you have.

When label catalog creation is in progress, a signal file **LabelCatalogUpdateInProgress.txt** is created in the same folder with the Automation configuration. This file prevents simultaneous generation of label catalog in an environment with multiple Automation servers. After the label catalog is generated, the signal file is deleted.

NiceLabel Automation saves the label catalog as an XML file **LabelCatalog.xml** in the same folder with the Automation configuration file.

The user account that you run your NiceLabel Automation service with must belong in the access role with the permissions to read folders and label templates to generate the label catalog.

Requirements:

- The user that runs Automation Service must have the permission to write to the DMS folder where Automation configuration is saved so that the signal file and label catalog file can be created in the same folder.
- Do not check in a file `LabelCatalog.xml` , or NiceLabel Automation will not be able to update it.

You can change the location for the signal file (by default, Automation saves it in the same folder, where you have stored the configuration file – MISX file):

1. Start **NiceLabel Automation Builder**.
2. Open the Automaton configuration (MISX file).
3. Edit the trigger **Label Catalog Async Updater**.
4. Select **Variables** tab.
5. Select **signalFile** variable.
6. Update the **Default value** to include the new path.

Requesting a list of variables from the label

To request a list of variables that are defined in the label, ABAP Package must set the **<VARIABLES />** XML element to **True** and provide the label name in the **<FORMAT />** XML element.

Loftware Automation reads the label variables and returns a list of variables with their properties. The list contains the “prompted” variables. These are the variables for which you provide values at print time. For example, “constants” or “database data sources” are not included in the list.

For the XML structure of the response, see chapter **Structure of <LabelVariables />** on page 98.

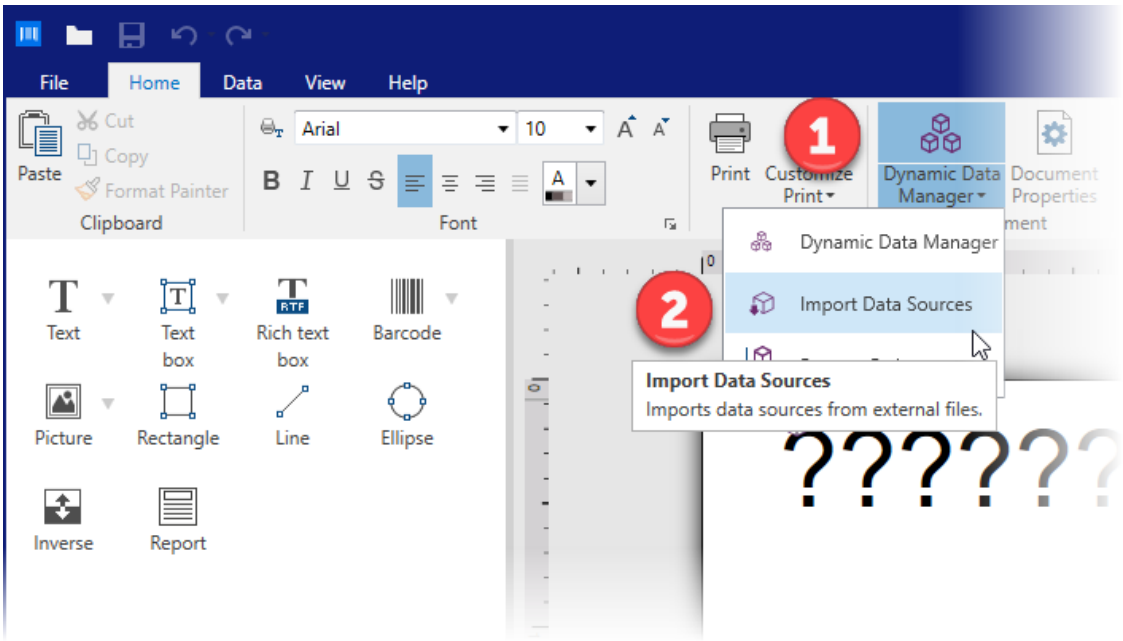
Generating a field catalog

The XML payload generated in the ABAP Package contains name-value pairs for the data points in the originating SAP transaction. These field names must have a matching counterpart in the label template. For each field name that you want to print in the label template, you must define a variable in the label template of the same name. Loftware Automation will then automatically map field values from XML payload to the label variable of the same name.

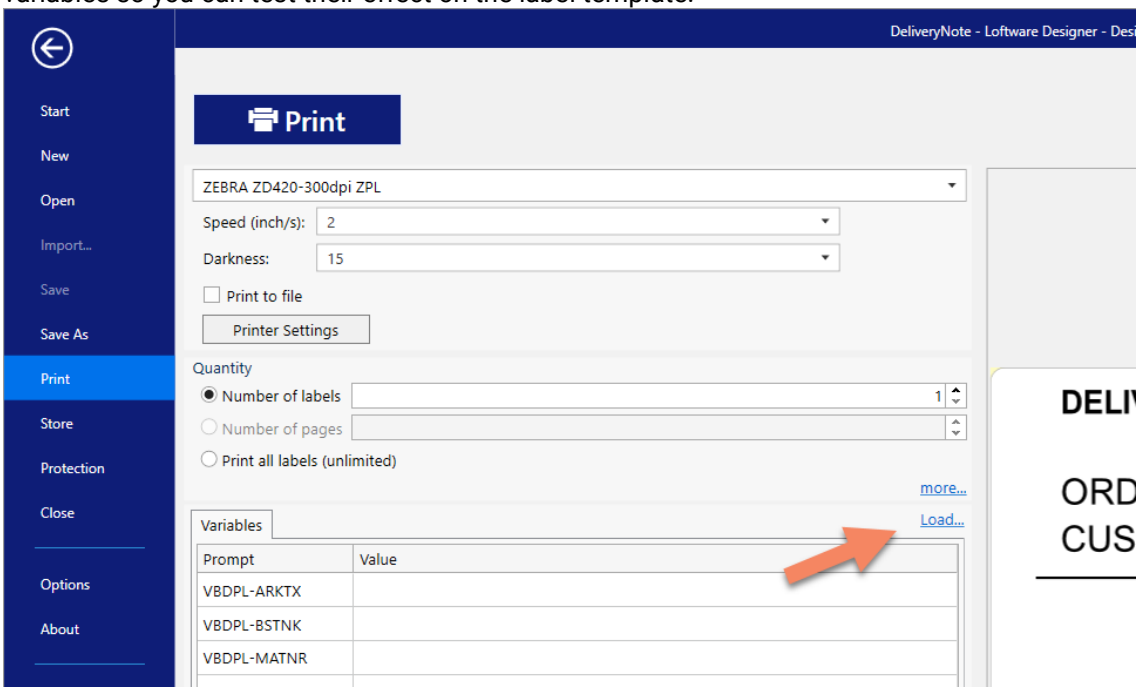
Loftware Automation exposes functionality to generate a field catalog for you. All name-value pairs provided in the XML payload are converted in the data file that a designer can import into the label template.

Loftware Automation can convert the name-value pairs from the XML payload into the following data structures to save time manually defining variables in the label template:

1. **Loftware “variable definition file” (file extension .NLVR).** When you import .NLVR file in the label template, all variables will be created automatically using the field names from the XML. The current values of fields from XML are used as “provisional values” – these are values that Loftware Designer will use during label design but are not default values for printing. You can import variables on the label template in **Dynamic Data Manager**.



2. **Loftware “values file” (file extension. VALUES).** You can browse for this file in the Print form when testing your label in Loftware Designer. Values from this file are temporarily applied to label variables so you can test their effect on the label template.



3. **CSV file (file extension .CSV).** This file is generated from a report section of the XML payload when such a section is found in XML. The report section provides repeatable data for shipping documents, like delivery notes, packing slips, invoices, and similar. The Report object requires a connection to a database source while you design it. You can use generated CSV as a database source.

To request a variable definition file, ABAP Package must set the **<DATAMODEL />** XML element to **True**.

All data files are stored in the same folder, where the Loftware Automation configuration file is stored.

You can change the location of the data files.

Permanently

1. Start **Loftware Automation Builder**.
2. Open the Automaton configuration (MISX file).
3. Edit the trigger you use: “HTTP (RFC”, “Web Service (SOA)”, or “Cloud Trigger”.
4. Go to the **Variables** tab.
5. Edit variable **_PATH_DATAFILES**.
6. For its default value enter the folder name, where you want to save the data files:
    a. **c:\folder** – the data files are saved in a folder “\folder” on a local drive C (on a computer where Automation is installed)
    b. **/folder** – the data files are saved in a folder “/folder” in the Document Management System to which Automation is assigned. You can also prepend the server name but is not necessary as Automation will add it on its own (e.g., https://server:8080/folder)
7. Save the configuration.

Dynamically with each request

1. Include the variable **_PATH_DATAFILES** inside the <Data> segment in the XML structure.
2. For its value send the folder name, where you want to save the data files:
    a. **c:\folder** – the data files are saved in a folder “\folder” on a local drive C (on a computer where Automation is installed)
    b. **/folder** – the data files are saved in a folder “/folder” in the Document Management System to which Automation is assigned. You can also prepend the server name but is not necessary as Automation will add it on its own (e.g., https://server:8080/folder)

Support for label variants

ABAP Package can use Loftware variant technology to prepare label variants for each label that is needed, down to the SKU level without having to manually create hundreds or thousands of label variations.

A label variant is a label template (.NLBL file) that has the data sources hard-coded in and is locked from editing. Loftware creates it out of the approved master label template and hard-codes the provided values for the variables defined in the label template. Each label variant will store data for a particular SKU. You cannot modify the label variant in Loftware Designer, the variant is not editable.

When creating label variants, you can choose to leave some of the variables unlocked. These are the data sources, for which you do not know the values upfront, like Batch/LOT numbers, production dates, or best-before dates.

To request the creation of label variants, ABAP Package must set the **<VARIANTS />** XML element to **True**. The names of variables that must remain unlocked in the generate variant file(s) are provided in **<VARIANTS_UNLOCKEDVARIABLES />** element as CSV list.

You can provide values for many variants in the same request to ABAP Package. Each **<Data />** segment provides data for one variant.

Each label variant is saved under its unique filename.
You can provide the variant file name with the value for key **_VARIANT_FILENAME**. You can use the same naming syntax as for the label names (in <FORMAT /> element inside Header). For more information, see chapter **Providing a label name** on page **67**.

```
...
        <VARIANTS>True</VARIANTS>
        <VARIANTS_UNLOCKEDVARIABLES>Batch</VARIANTS_UNLOCKEDVARIABLES>
    </Header>
    <Data>
        <Item Id="_VARIANT_FILENAME" Position="000000" Type="">/Production-
AS123/LabelVariants/3847638-234.nlbl</Item>
...
```

If required, the variant file name can also be auto-generated out of the other key-value pairs provided in the <Data/> segment for each variant. In this case, you can modify the Automation configuration to add logic to generating the variant file name. Add your actions inside the placeholder already reserved for such a case. Open the Automation configuration and see the action **3.5.1.4.1.1.2.5.2 Custom variant name (placeholder)**.

Exporting data

You can use the method EXPORT_DATA to run custom actions in Automation to save the data from SAP to some intermediate database or other resources. Many SAP integration projects use the ABAP Package not for label printing but for exporting the data out of SAP. The actual print is initiated from the custom form application running in Loftware GUI.

In the previous versions of the ABAP Package, you had to change the Automation configuration of the existing PRINT method to implement saving the data instead of printing. From this version, the method for exporting data is built into the Automation configuration.

The Automation configuration has all the groundwork prepared. It contains the placeholder that will execute when you set the **<EXPORT_DATA />** XLM element to **True**. The placeholder comes without any defined actions. You are free to add your custom actions and adapt the configuration to your specific need and environment.

Communication extendibility

Often, the customers use the ABAP Package outside its intended usage. For example, they would use it to export the data from SAP and not use printing functionality at all. Because you can freely modify the Automation configuration it is easy to extend it with new custom actions.

To help you with the modification even further, ABAP Package supports up to three custom XML fields in the Header part of the XML message. You can use fields CUSTOM1, CUSTOM2, and CUSOTOM3 as Boolean flags to run your action workflows. On the Automation side, the configuration contains the matching three placeholders, where you would define your customs actions.

For example, when you set the value of the CUSTOM1 element to True, the actions in the matching placeholder CUSTOM1 will execute.

The XML response coming from Automation back to the ABAP Package has also been expanded with new custom fields (these are optional). You can include your custom messages in the feedback message if you need to provide more data back to the ABAP Package. The ABAP Package will parse these custom messages, if they exist, and make them available in your SAP transaction.

Configuring transaction to call ABAP Package

When a specific event happens in the SAP transaction, and label printing is required, the transaction must send the data into ABAP Package. The ABAP Package uses the enhancement spots for supported standard SAP transactions (e.g., outbound delivery) for which label printing is necessary. For these supported transactions, label printing executes out-of-the-box. You do not have to code anything in your SAP system. For details, see chapter **Enhancement spots**.

If you want to print labels from other transactions or as a part of specific events, you must configure SAP to communicate with the ABAP Package by yourself. Depending on the requirement and the process, you can use one of the following approaches to send the data from the SAP application to Loftware Automation using the ABAP Package:

- User exit in SAP standard coding.
- Business add-in (BAdI) / Enhancement.
- Create document output and change the print program.
- Custom ABAP program.

If customers integrate label printing with the SAP standard process (for example upon creating a delivery note), the preferred integration method is to use a custom **output** on the delivery object. The reason for this is that such a method also supports execution logging to some extent and is straightforward to set it up.

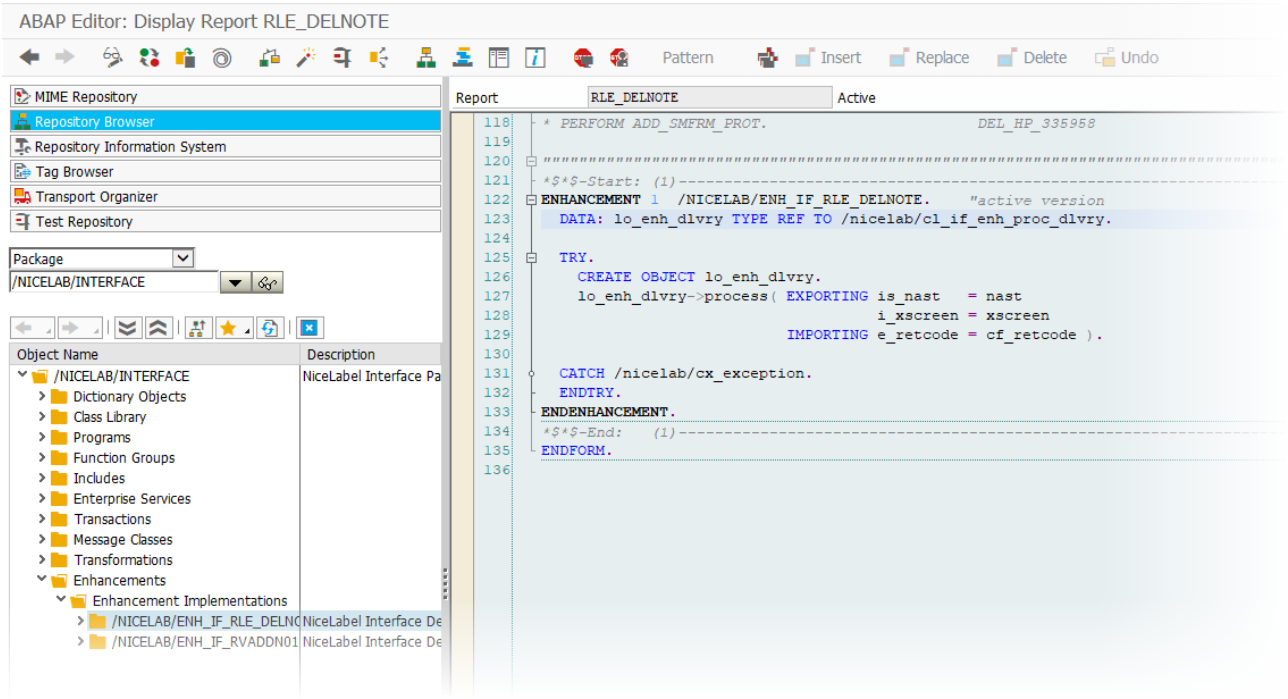
If there is a requirement to implement label printing in a process during which a standard **output** is not possible, consider the option for using a user-exit, BAdI, or enhancement spots.

Enhancement spots

Among the SAP software advantages is the possibility to adapt the software to your requirements, and the possibility to keep the adaptations during the upgrade. Enhancement spots allow you to add custom functionalities to the standard SAP programs.

ABAP Package extends the functionality of standard print programs. For example, when the user issues an outbound delivery and prints or previews the standard form, at the same time the enhancement spot in ABAP Package activates and sends the same data into Loftware Automation for processing. SAP prints the standard form and Loftware Automation prints the labels.

Using the enhancement spots technology, the ABAP Package provides no-coding out-of-the-box label printing functionality to the supported standard SAP applications. The usage of enhancement spots can be enabled or disabled in the configuration table (see chapter **Enabling enhancement spots** on page 35).



Example for outbound deliveries (VL02n)

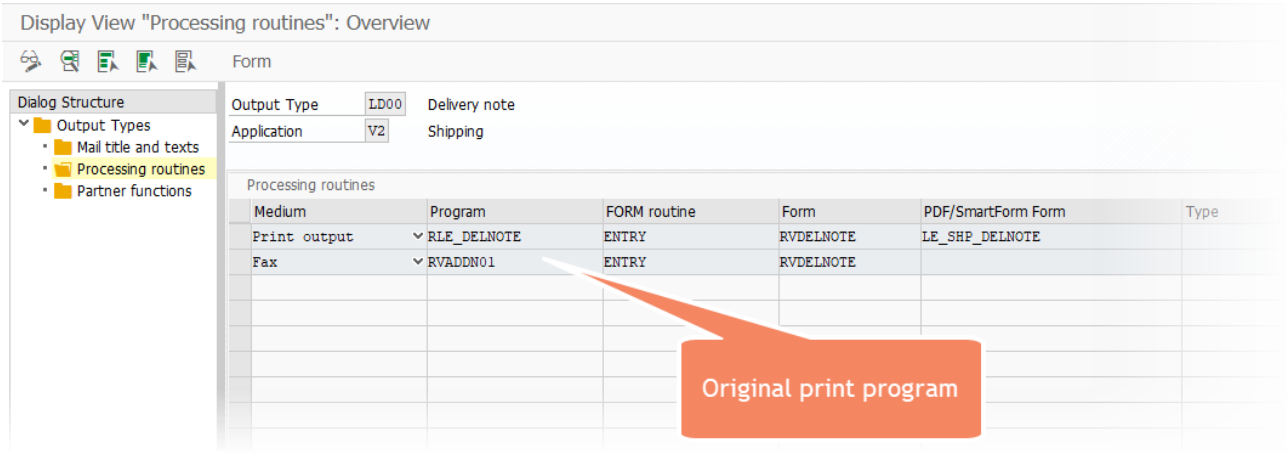
When using the standard output approach, the standard print program (which normally delivers data into SAP Spooler) must change in a way to deliver data into ABAP Package. First, you must make a copy of the standard output print program that the current/default output type uses and modify it to communicate with ABAP Package.

When done, you must create a new output type and use the updated print program with it. To print labels, a user selects the new output type or configures the print program to execute the output automatically upon *object (*delivery in our example*)* creation. Programmable changes are necessary for the ABAP code.

An example of ABAP Package integration is demonstrated on the outbound delivery transaction (VL02n) using a new **document output**. This transaction is used for “deliveries” – for labels containing data for outbound delivery.

ABAP Package includes the custom print program for VL02n.

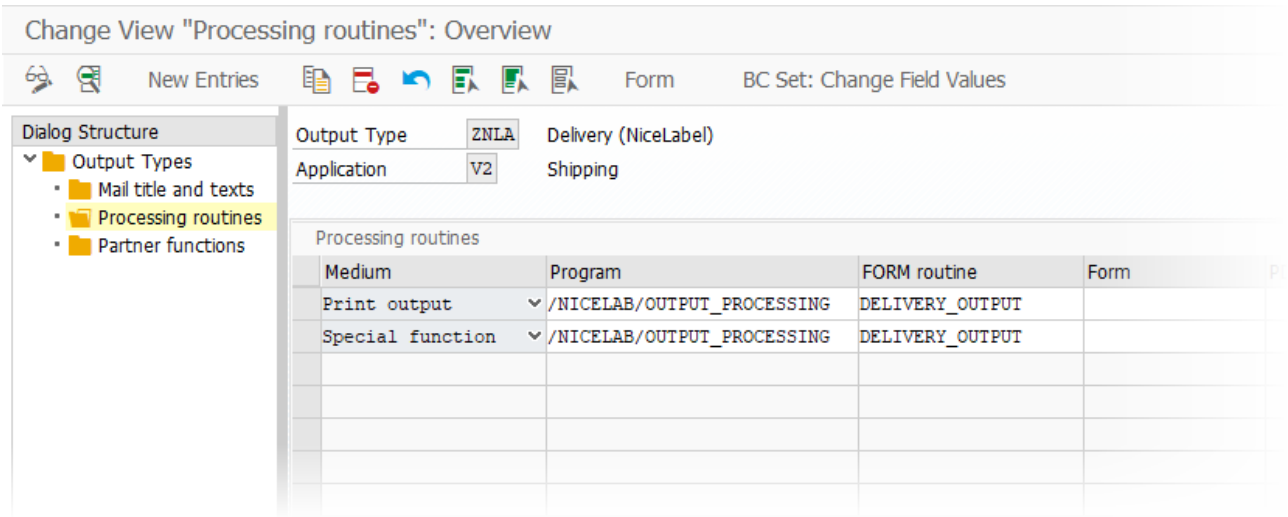
By default, the transaction VL02n uses the output type **LD00**. If you drill down to the processing routines defined for this output type in the NACE transaction code, you would learn that one of the transmission medium options is configured to use the **RLE_DELNOTE / RVADDN01** print programs.



Creating a new output type

To redirect data output to ABAP Package:

1. In the NACE transaction code, create a new output type. In this example, the **ZNLA** output type is created.
    a. Launch the transaction code **NACE**.
    b. For Shipping (where outbound delivery fits into), select the **V2** application and click **Output types**.
    c. Change display mode to edit mode and click **New Entries**.
    d. In the **General data** tab, fill ID of **Output Type**, description, and access type.
    e. In the **Default values** tab, set **Dispatch time** to when you want the message to be processed.
    f. In the **Default values** tab, set message **Transmission Medium** to Special function.
    g. In **Processing Routines**, you must configure the program name for each routine. For **Program**, select enter **/NICELAB/OUTPUT_PROCESSING**. For **Form Routine**, select **DELIVERY_OUTPUT**.
    h. Save changes.

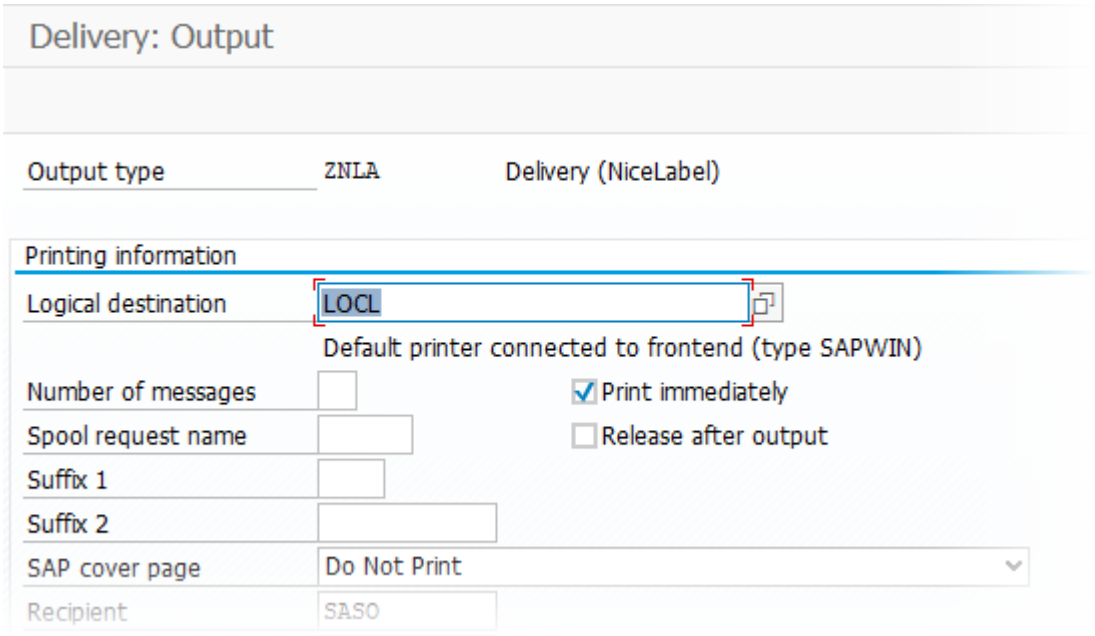


Using the new output type for processing

To use the new output type in the VL02n transaction code, do the following:

1. Run **VL02n** transaction code.
2. Type **ID** of the outbound delivery message and confirm it with <Enter>.
3. In **SAP GUI**, select **Extras>Delivery Output>Header**.

4. For new output item, type **ZNLA** for the Output type.
5. Type the language (EN in this example).
6. In the **Communication method**, select **LOCL** for the Logical destination.
   Enable **Print immediately** check box.



7. Save the message.

When you save the output, the custom print program maps the current delivery data (LIKP, LIPS structures) with ABAP Package. This creates an XML message and sends it for processing to the Loftware Automation system. Feedback about the processing is saved in the output **Processing log**.

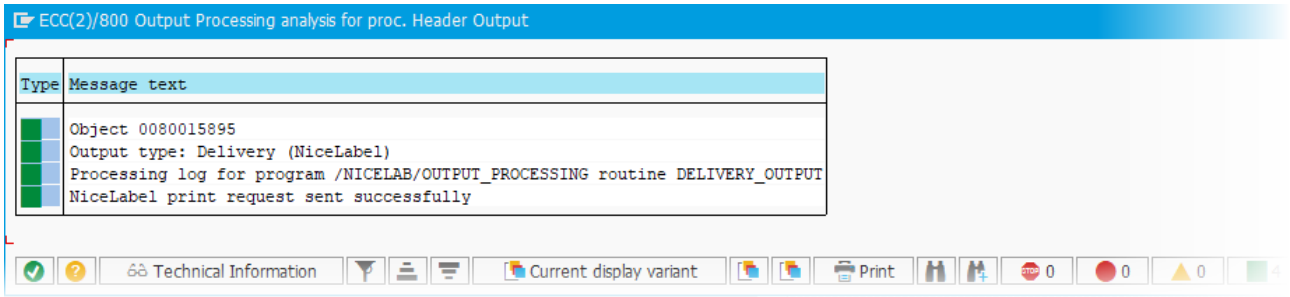


Figure 5: Feedback from Loftware Automation about data processing

Label sample with predefined data sources

Sample label for outbound delivery is included in ABAP Package. The sample label contains all SAP data sources from outbound delivery (VBDPL-*). You can import data sources from the label to any new label you create and consequently save time.

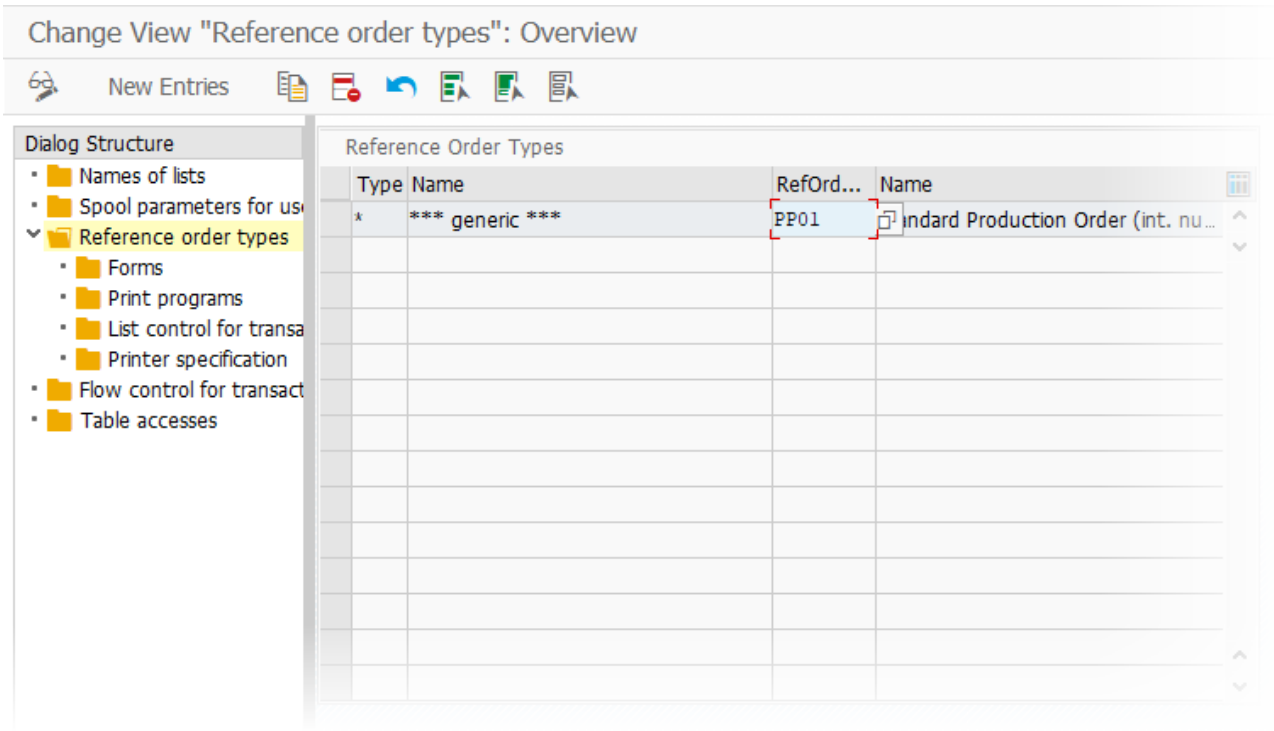
Look for label **`DeliveryNote.nlbl`**.

Example for production order (CO02)

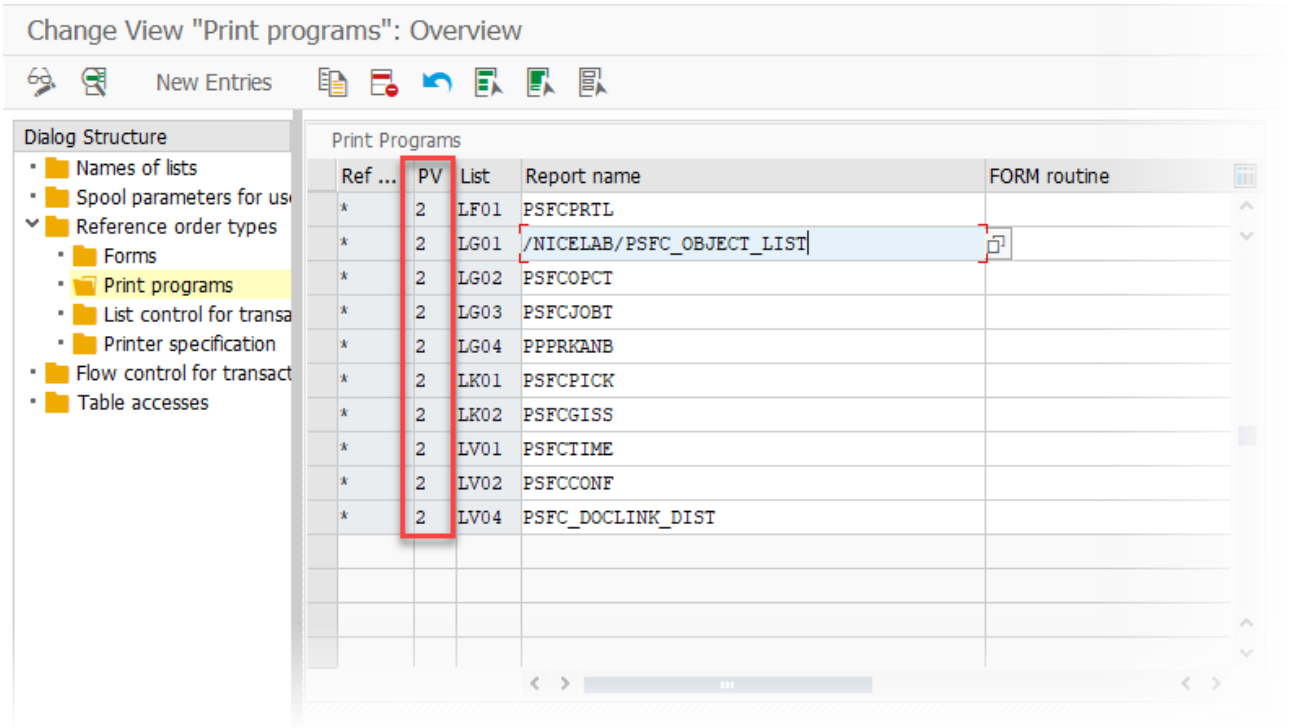
This section describes the necessary customizing steps for setting up the basic print functionality of the production order object list. These steps include the print program provided with ABAP Package. Order type(s) used in the screenshots below are used for example purposes only. It is up to the customer needs to configure printing for the correct order type, plant, or MRP group.

Configuring the system

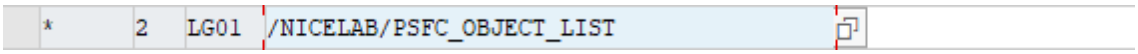
1. In **SAP GUI**, run tcode **OPK8** – Maintain print control of production orders.
2. Select the order type:



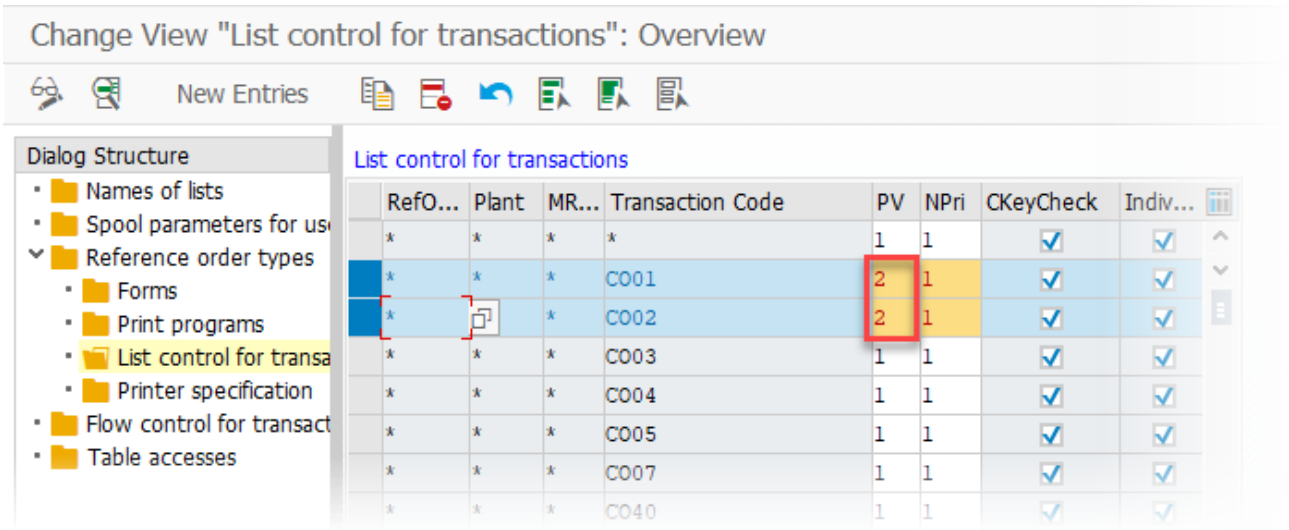
3. Go to the **Print programs** node and define a new print variant (PV). Take the existing variant as a template and set the print variant number (in this example, the new print variant 2 is defined).



4. In the new variant, set the provided print program **/NICELAB/PSFC_OBJECT_LIST** for list **LG01**.



5. Go to **List control for transaction** node and configure the print variant on **CO01 / CO02** transaction level.

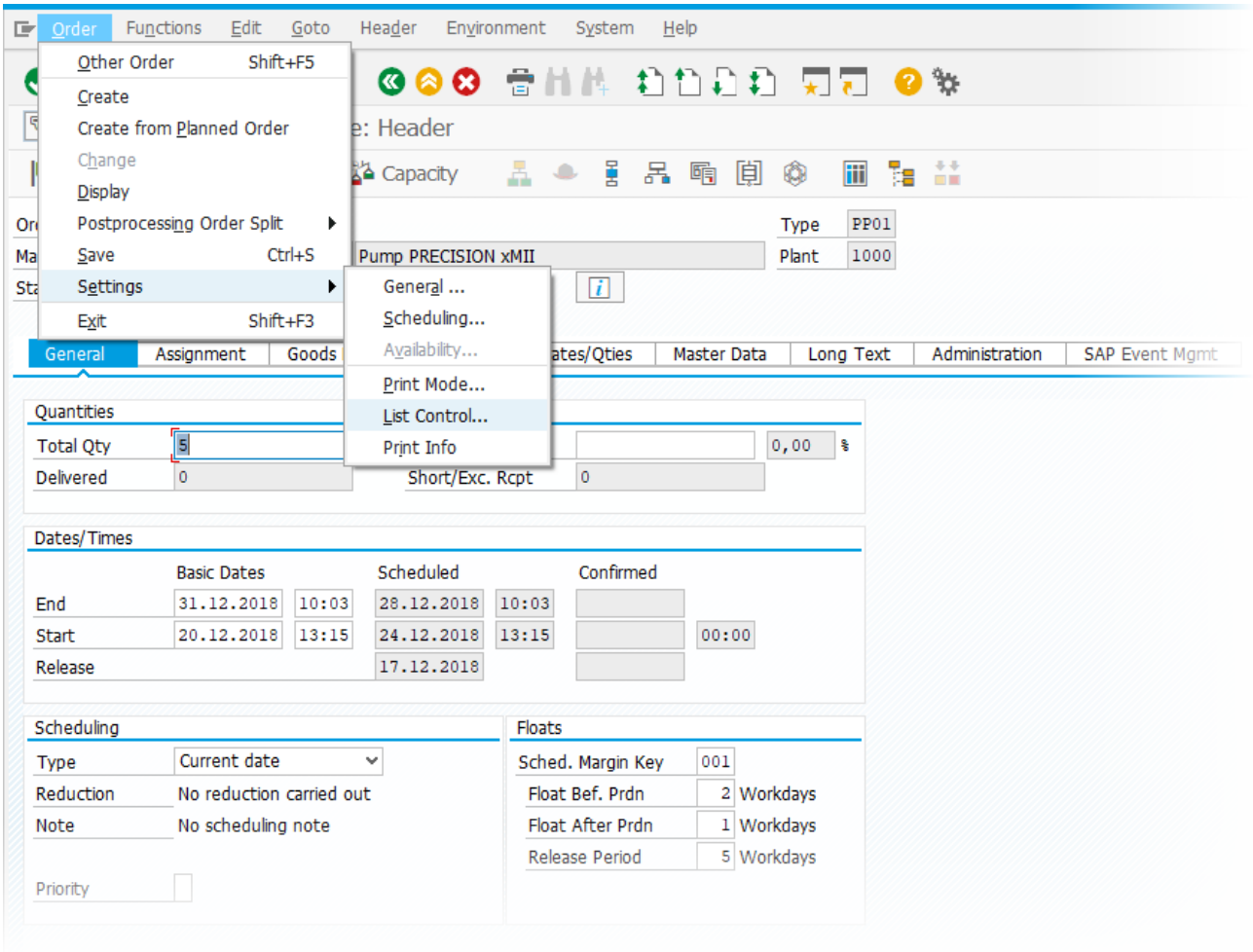


Printing labels

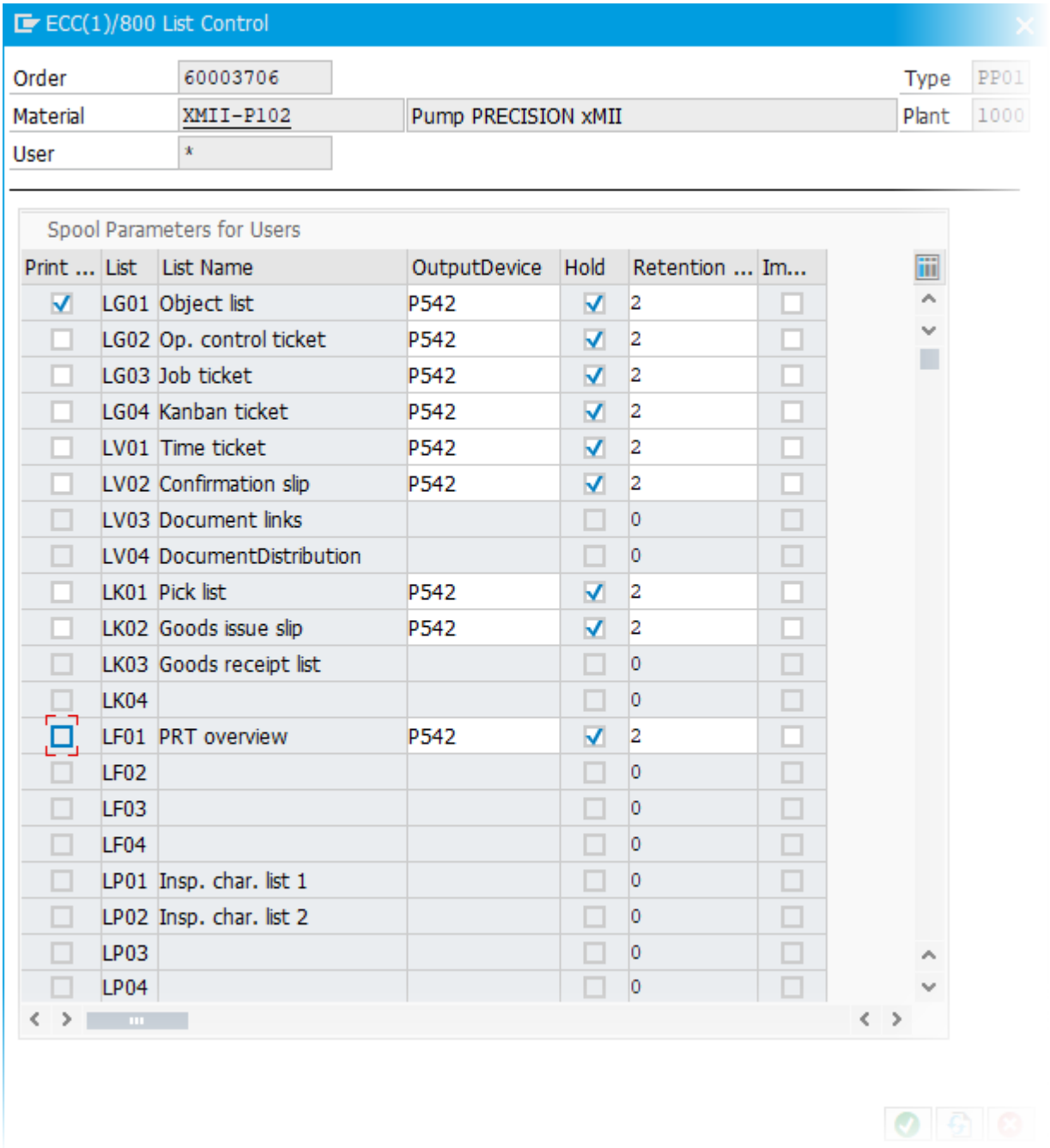
NOTE: The example below assumes execution on a specific SAP server with the available sample data. In your environment, the steps are the same, but the data varies.

1. In **SAP GUI**, run tcode **CO02**.
2. For **Order** number, type **60003706** and press Enter.

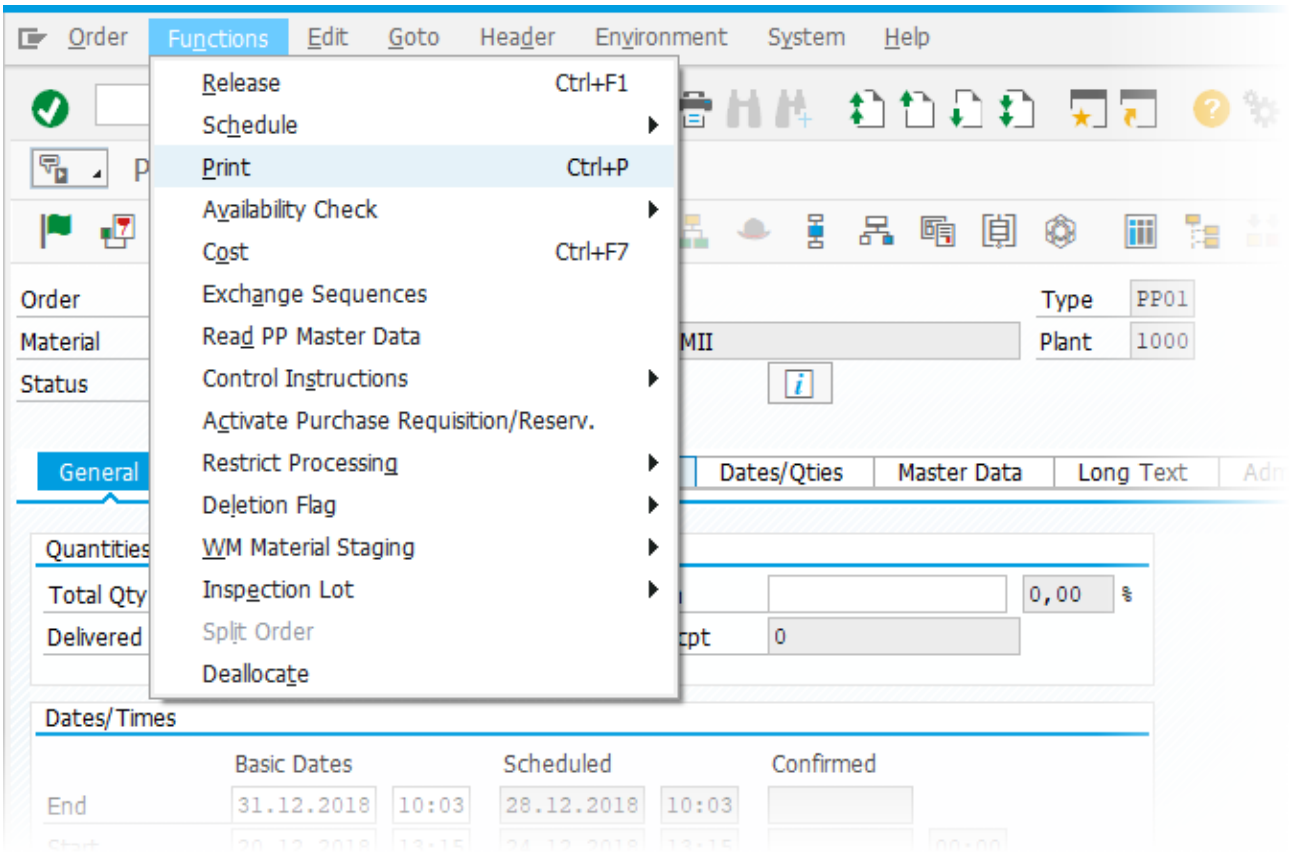
3. Open **Order > Settings > List control**.



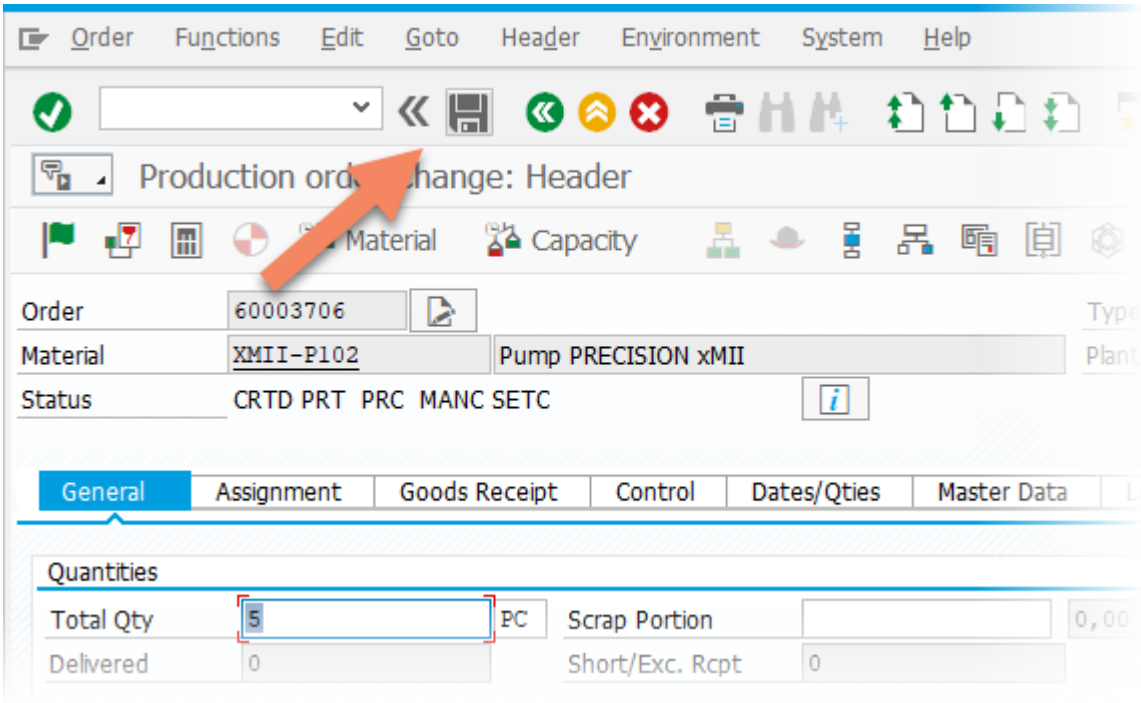
4. Uncheck all unnecessary lists, leave just **LG01** selected and click **Continue** button  or press **Enter**.


ECC(1)/800 List Control
Order
60003706
Type
PP01
Material
XMII-P102
Pump PRECISION xMII
Plant
1000
User
*
Spool Parameters for Users
Print ... List List Name OutputDevice Hold Retention ... Im...
LG01 Object list P542 2
LG02 Op. control ticket P542 2
LG03 Job ticket P542 2
LG04 Kanban ticket P542 2
LV01 Time ticket P542 2
LV02 Confirmation slip P542 2
LV03 Document links 0
LV04 DocumentDistribution 0
LK01 Pick list P542 2
LK02 Goods issue slip P542 2
LK03 Goods receipt list 0
LK04 0
LF01 PRT overview P542 2
LF02 0
LF03 0
LF04 0
LP01 Insp. char. list 1 0
LP02 Insp. char. list 2 0
LP03 0
LP04 0

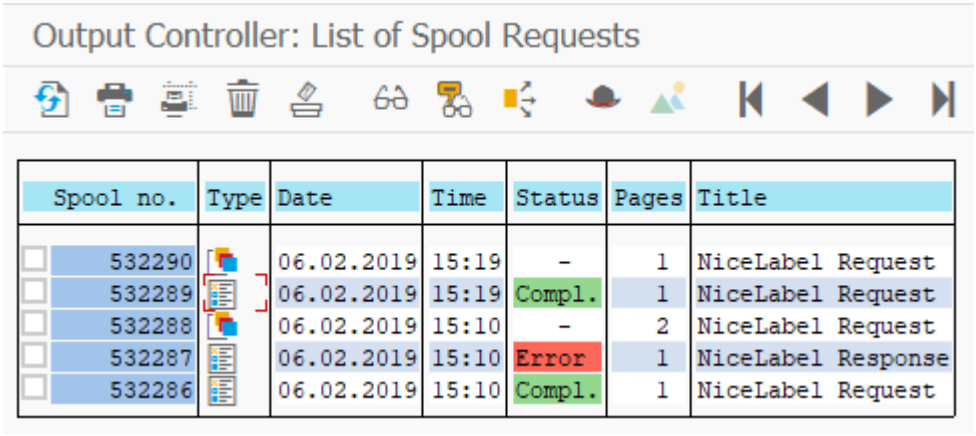
5. Select **Functions > Print** (or press **Ctrl+P**).



6. Click the Save button or press **Ctrl+S**. ABAP Package sends a print request to the Loftware integration server.



7. Run tcode SP01 and verify the result.



Label sample with predefined data sources

The sample label for outbound delivery is included in the ABAP Package. This label contains all SAP data sources from the production order (CAUFVD_P-*). You can import the data sources from this label to any newly created label to save your time.

Look for label **`ProductionOrder.nlbl`**.

ABAP code examples

This chapter provides ABAP code snippets from the programs that are delivered with the ABAP Package to better understand how you should establish communication from your ABAP programs with the ABAP Package.

Creating basic print request

In the below coding we will demonstrate steps that are needed to generate a simple print request to Loftware, which will contain two variables - MATERIAL and DESCRIPTION.

```
DATA: ls_request    TYPE /nicelab/rqst_data_out,      "Print Request structure
      ls_nl_label   LIKE LINE OF ls_request-labels,
      ls_nl_item_t  TYPE /nicelab/rqst_item_data_tt,
      ls_nl_item    TYPE /nicelab/rqst_item_data.
DATA: lv_msg        TYPE string.
DATA: lo_nl_api TYPE REF TO /nicelab/cl_interface_demo.

CLEAR: ls_request, ls_nl_label.
"fill print request header
ls_nl_label-header-format         = 'Label.nlbl'.         "Name of the label in Loftware Automation
ls_nl_label-header-preview_format = /nicelab/cl_if_util=>co_resp_format_pdf.
ls_nl_label-header-quantity       = 1.
ls_nl_label-header-printername    = 'NiceLabel_Printer'.  "Name of the printer in Loftware
Automation
ls_nl_label-header-print          = /nicelab/cl_if_util=>bool_2_tf( abap_true ).

"fill request items (variables)
CLEAR ls_nl_item.
ls_nl_item-item_id    = 'MATERIAL'.
ls_nl_item-item_value = '12345678'.
```

```
APPEND ls_nl_item TO ls_nl_item_t.
CLEAR ls_nl_item.
ls_nl_item-item_id    = 'DESCRIPTION'.
ls_nl_item-item_value = 'Demo Material'.
APPEND ls_nl_item TO ls_nl_item_t.

APPEND ls_nl_item_t TO ls_nl_label-data.
APPEND ls_nl_label TO ls_request-labels.

"create Loftware API object instance
CREATE OBJECT lo_nl_api.

"send a print request to Loftware
CALL METHOD lo_nl_api->do_print_xml
  EXPORTING
    i_nl_data   = ls_request
  EXCEPTIONS
    print_error = 1
    OTHERS      = 2.

IF NOT sy-subrc IS INITIAL.
  "get_response error
  CLEAR lv_msg.
  lo_nl_api->response_error_get( EXPORTING i_dialog     = abap_false
                                 IMPORTING e_error_desc = lv_msg ).
ENDIF.
```

Dynamically defining the endpoint

The ABAP Package always communicates with the Automation endpoint that receives the payload for label printing.

- When you have just one Automation installed, you can define its endpoint in the configuration table with the transaction **/NICELAB/IF_CTRL**.
- When you have multiple Automation servers, you must tell the ABAP Package which one should receive the data at each execution. You can dynamically define the valid endpoint for each call of the ABAP Package.

This sample ABAP code iterates through the list of endpoints and applies the endpoint to the ABAP Package call. You can fill the list with available endpoint values that you maintain in one of your tables.

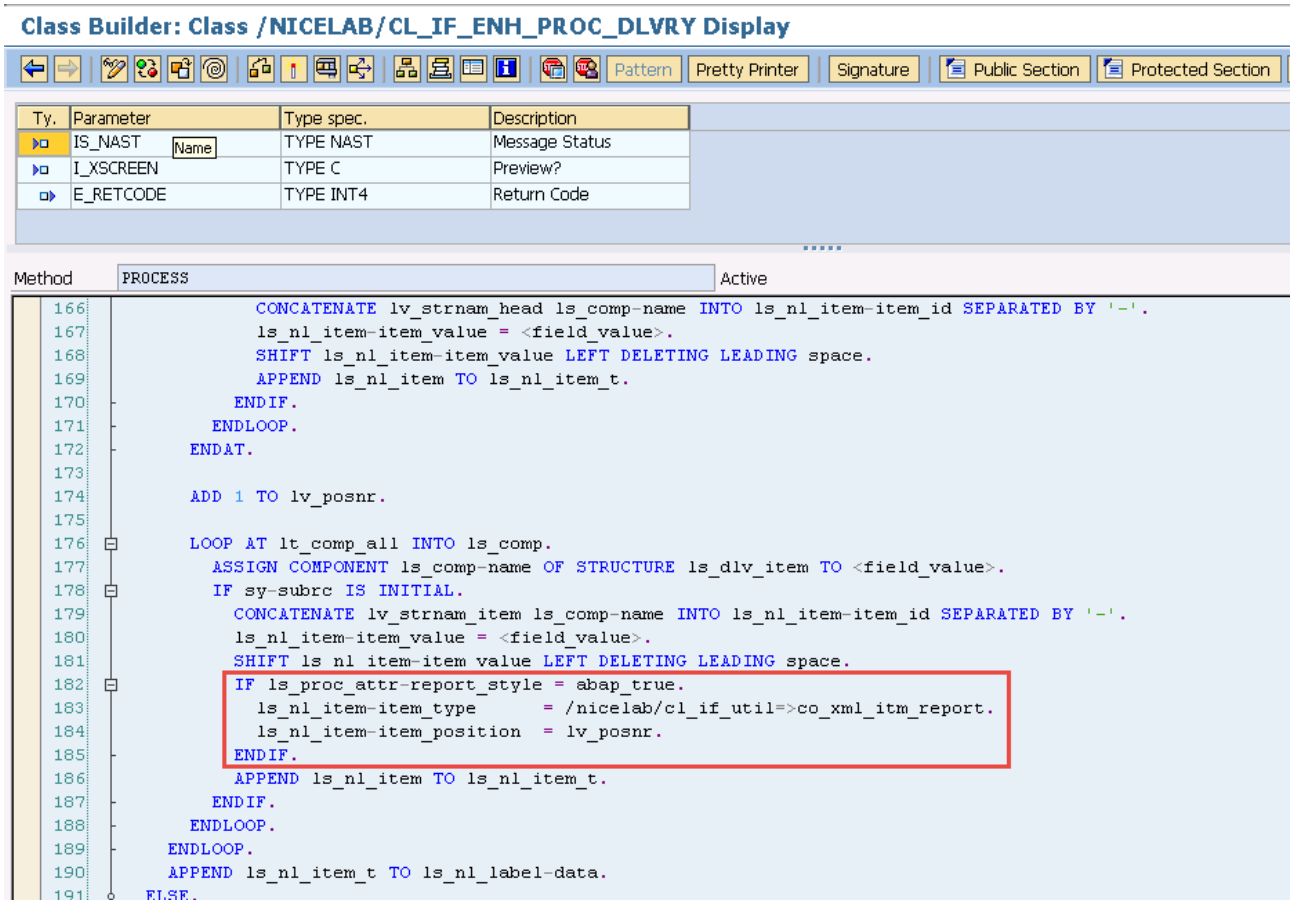
```
REPORT  znicelab_multi_endpoint_sample.


* Loftware Interface - API - instance
DATA: go_nicelabel  TYPE REF TO /nicelab/cl_interface_demo.

* Loftware Request
DATA: gs_nl_data     TYPE /nicelab/rqst_data_out.

" Loftware operation mode
DATA: gt_nl_operation TYPE TABLE OF /nicelab/if_operation.
```

```
DATA: gs_nl_operation LIKE LINE OF gt_nl_operation.


"internal table with endpoint list
CLEAR gs_nl_operation.
gs_nl_operation-operation_mode  = /nicelab/cl_if_util=>co_op_mode_http.
gs_nl_operation-rfc             = 'NL_RFC01'.
APPEND gs_nl_operation TO gt_nl_operation.
gs_nl_operation-operation_mode  = /nicelab/cl_if_util=>co_op_mode_http.
gs_nl_operation-rfc             = 'NL_RFC02'.
APPEND gs_nl_operation TO gt_nl_operation.
gs_nl_operation-operation_mode  = /nicelab/cl_if_util=>co_op_mode_http.
gs_nl_operation-rfc             = 'NL_RFC03'.
APPEND gs_nl_operation TO gt_nl_operation.

"process endpoint list
LOOP AT gt_nl_operation INTO gs_nl_operation.

  FREE: go_nicelabel, gs_nl_data.

  "create API instance with operation mode
  CREATE OBJECT go_nicelabel
    EXPORTING
      i_operation_mode = gs_nl_operation-operation_mode.

  "set endpoint name according to type (RFC/SOA)
  CASE gs_nl_operation-operation_mode.

    WHEN /nicelab/cl_if_util=>co_op_mode_soa.

      go_nicelabel->logical_port_set( gs_nl_operation-logical_port ).

    WHEN /nicelab/cl_if_util=>co_op_mode_http.

      go_nicelabel->rfc_destination_set( gs_nl_operation-rfc ).

    WHEN OTHERS.
      "nothing
      CONTINUE.

  ENDCASE.

  "fill Loftware request structure
*  gs_nl_data-....


  "send request to Loftware
  CALL METHOD go_nicelabel->do_print_xml
    EXPORTING
      i_nl_data    = gs_nl_data
    EXCEPTIONS
      print_error = 1
```

```
      OTHERS       = 2.
  IF sy-subrc <> 0.
  ENDIF.

ENDLOOP.
```

Creating *report style* print request

Below code in /NICELAB/CL_IF_ENH_PROC_DLVRY (-> PROCESS) class demonstrates how to generate and send **Report Style** print request to Loftware. In order to generate a report style label, you must fill ***item_type*** attribute in the items segment with constant "Report" (re-use constant`/nicelab/cl_if_util=>co_xml_itm_report`). Additionally, you must fill ***item_position*** with the respective line item position. All items (variables) that have the same position value belong to the same line item section in the report label (the same row).



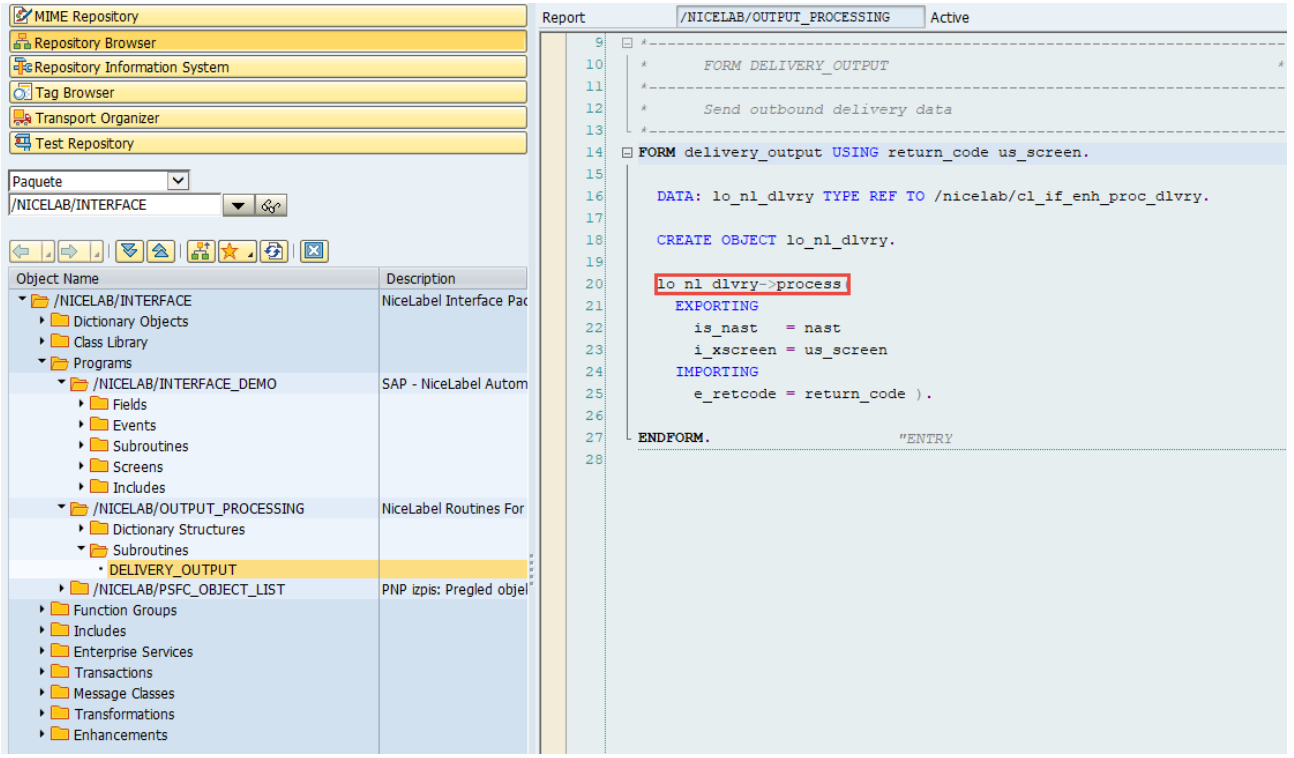
Outbound delivery

Look at the program we provide for outbound delivery.

Do the following

1. In **SAP GUI**, run tcode **SE80**.
2. Navigate to the program **/NICELAB/OUTPUT_PROCESSING**.

3. See routine **DELIVERY_OUTPUT** and method **lo_nl_dlvry->process**.



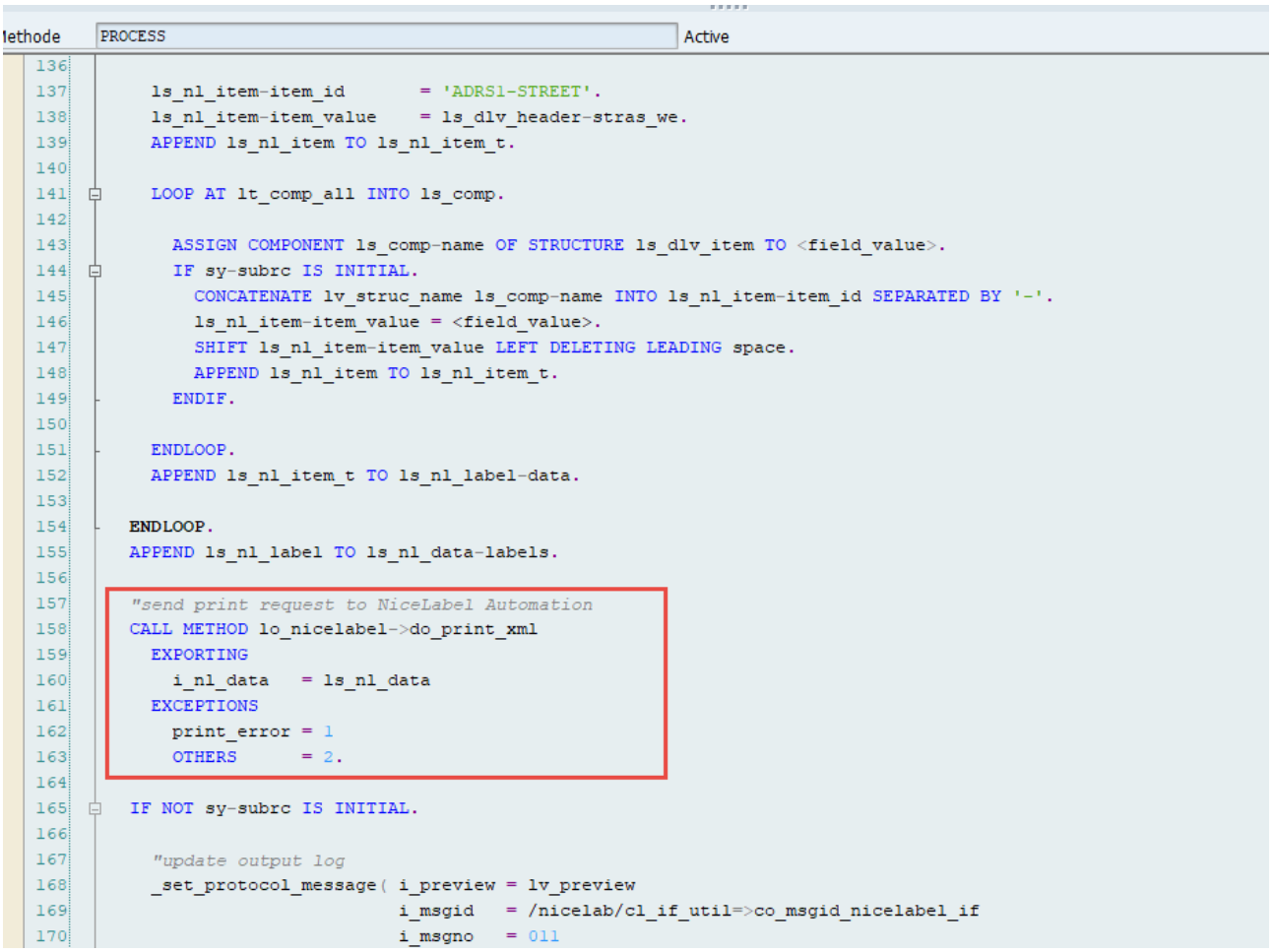
4. This method contains an example, where we use SAP function **RV_DELIVERY_PRINT_VIEW** to read data for items from the delivery, loop through **LT_DLV_ITEM** items, and fill data into **ls_nl_item_t** table that contains name-value pairs.

Ty.	Parameter	Type spec.	Description
▶□	IS_NAST	TYPE NAST	Message Status
▶□	I_XSCREEN	TYPE C	Preview?
□▶	E_RETCODE	TYPE INT4	Return Code

Methode PROCESS Active

```
103     ls_comwa-vbeln = is_nast-objky.
104     ls_comwa-kunde = is_nast-parnr.
105     ls_comwa-parvw = is_nast-parvw.
106
107     "get outbounbd delivery data
108     CALL FUNCTION 'RV_DELIVERY_PRINT_VIEW'
109       EXPORTING
110         comwa = ls_comwa
111       IMPORTING
112         kopf  = ls_dlv_header
113       TABLES
114         pos   = lt_dlv_item.
115
116     "get components of structure
117     lo_rtti_struc ?= cl_abap_structdescr=>describe_by_data( ls_dlv_item ).
118     lt_comp = lo_rtti_struc->get_components( ).
119     lv_struc_name = lo_rtti_struc->get_relative_name( ).
120
121     _fill_nl_item( EXPORTING it_comp = lt_comp
122                    CHANGING ct_comp = lt_comp_all ).
123
124     LOOP AT lt_dlv_item INTO ls_dlv_item.
125
126       REFRESH ls_nl_item_t.
127       CLEAR ls_nl_item.
128
129       ls_nl_item-item_id       = 'ADRS1-CITY1'.
130       ls_nl_item-item_value    = ls_dlv_header-ort01_we.
131       APPEND ls_nl_item TO ls_nl_item_t.
132
133       ls_nl_item-item_id       = 'ADRS1-NAME1'.
134       ls_nl_item-item_value    = ls_dlv_header-name1_we.
135       APPEND ls_nl_item TO ls_nl_item_t.
136
137       ls_nl_item-item_id       = 'ADRS1-STREET'.
138       ls_nl_item-item_value    = ls_dlv_header-stras_we.
139       APPEND ls_nl_item TO ls_nl_item_t.
140
141       LOOP AT lt_comp_all INTO ls_comp.
142
143         ASSIGN COMPONENT ls_comp-name OF STRUCTURE ls_dlv_item TO <field_value>.
```

5. Next, we send the complete envelope into the ABAP Package.



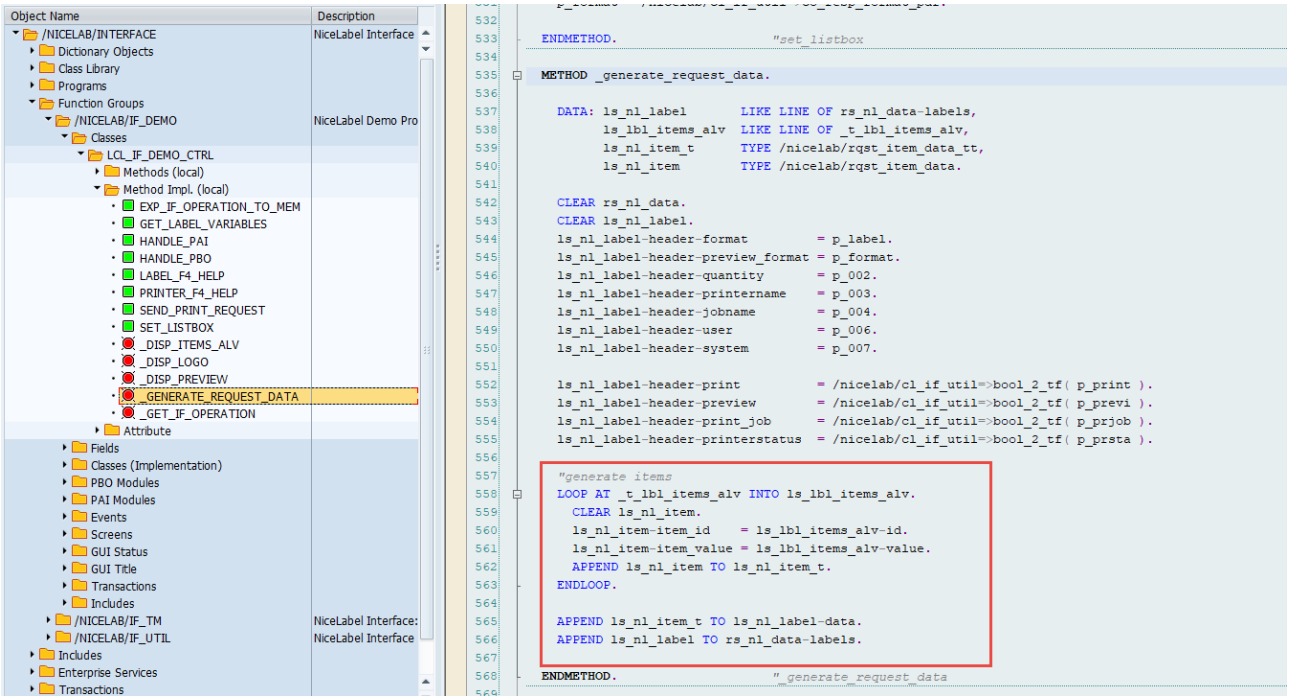
```
lethode   PROCESS                                                    Active
136
137       ls_nl_item-item_id       = 'ADRS1-STREET'.
138       ls_nl_item-item_value    = ls_dlv_header-stras_we.
139       APPEND ls_nl_item TO ls_nl_item_t.
140
141       LOOP AT lt_comp_all INTO ls_comp.
142
143         ASSIGN COMPONENT ls_comp-name OF STRUCTURE ls_dlv_item TO <field_value>.
144         IF sy-subrc IS INITIAL.
145           CONCATENATE lv_struc_name ls_comp-name INTO ls_nl_item-item_id SEPARATED BY '-'.
146           ls_nl_item-item_value = <field_value>.
147           SHIFT ls_nl_item-item_value LEFT DELETING LEADING space.
148           APPEND ls_nl_item TO ls_nl_item_t.
149         ENDIF.
150
151       ENDLOOP.
152       APPEND ls_nl_item_t TO ls_nl_label-data.
153
154     ENDLOOP.
155     APPEND ls_nl_label TO ls_nl_data-labels.
156
157     "send print request to NiceLabel Automation
158     CALL METHOD lo_nicelabel->do_print_xml
159       EXPORTING
160         i_nl_data   = ls_nl_data
161       EXCEPTIONS
162         print_error = 1
163         OTHERS      = 2.
164
165     IF NOT sy-subrc IS INITIAL.
166
167       "update output log
168       _set_protocol_message( i_preview = lv_preview
169                              i_msgid   = /nicelab/cl_if_util=>co_msgid_nicelabel_if
170                              i_msgno   = 011
```

Demo application

We use a similar approach (as for the outbound delivery) also in the demo application.

Do the following:

1. In **SAP GUI**, run tcode **SE80**.
2. Navigate to **_GENERATE_REQUEST_DATA** as shown in the screenshot below.
3. The transfer of the name-value pairs (displayed in the table object on-screen) executes in the **_GENERATE_REQUEST_DATA** method and data is sent out when executing the **SEND_PRINT_REQUEST** method.



Print programs

The ABAP Package delivers the following print programs to be used with the respective standard SAP transactions. You can study these programs to see how we collect data (key-value pairs) and send them into the ABAP Package API.

The provided print programs:

- **/NICELAB/OUTPUT_PROCESSING**. For Outbound Delivery (VL02n).
- **/NICELAB/PSFC_OBJECT_LIST**. For Production Order (CO02).

Data exchange

This topic describes the structure of messages transferred between ABAP Package and Loftware Automation, and the structure of Loftware Automation feedback.

Structure of XML data sent from ABAP Package

The XML as created by ABAP Package encapsulates the data and methods you are executing with Loftware Automation. Each request from ABAP Package to Loftware Automation must include at least one method but can contain many. For example, you can execute label preview and label print in the same request. The methods you want to execute with Loftware Automation are configured with Boolean data types in the XML “Header” part.

ABAP Package encapsulates the data received from the SAP application into the XML structure in two parts.

- **Header.** This element contains meta-data information, such as which method you want to execute in Automation, which label to use, which printer to use, how many labels to print, etc. Frequently used fields are FORMAT, QUANTITY, PRINTERNAME, JOBNAME, PRINT, PREVIEW, and PRINTERSTATUS. You can include just the fields that are required for a specific API call. E.g., if you want to print a label, you would include just FORMAT and PRINT.

  There can be just one **Header** element.

- **Data.** This element contains name-value pairs for data fields in SAP applications. You would provide the name and value for all fields you want to use on a label. You can provide as many name-value pairs as you need, there is no limit on the number of items. The field name is provided in the attribute **Id**, the field value is provided in the **element value**.

  There can be many **Data** elements, each providing data for a new label or report.

Sample XML file:

```
<?xml version="1.0"?>
<LABELS>
    <Header>
        <FORMAT>label.nlbl</FORMAT>
        <QUANTITY>1</QUANTITY>
        <QUANTITY_IDENTICALCOPIES />
        <PRINTERNAME />
        <TRAY />
        <PRINTERSTATUS>False</PRINTERSTATUS>
        <JOBNAME />
        <USER>USER</USER>
        <SYSTEM>ECC</SYSTEM>
        <SYSTEM_TYPE>SAP</SYSTEM_TYPE>
        <PRINT>False</PRINT>
        <PREVIEW>True</PREVIEW>
        <PREVIEW_FORMAT>PDF</PREVIEW_FORMAT>
        <EMAIL_RECIPIENTS/
        <PRINTERS>False</PRINTERS>
```

```
        <PRINT_JOB>False</PRINT_JOB>
        <VARIABLES>False</VARIABLES>
        <CATALOG>False</CATALOG>
        <CATALOG_ROOT>/Labels</CATALOG_ROOT>
        <DATAMODEL />
        <CLOUDPRINT />
        <CLOUD_APIKEY />
        <CLOUD_APIVERSION />
        <CLOUD_LABELVERSION />
        <PRINTERSETTINGS />
        <LABELSETTINGS />
        <VARIANTS />
        <VARIANTS_UNLOCKEDVARIABLES />
        <EXPORT_DATA />
        <CUSTOM1 />
        <CUSTOM2 />
        <CUSTOM3 />
    </Header>
    <Data>
        <Item Id="FIELD1">Loftware</Item>
        <Item Id="FIELD2">SAMPLE</Item>
        <Item Id="FIELD3">12345</Item>
        <Item Id="FIELD4">12345</Item>
        <Item Id="FIELD5">123456789012</Item>
    </Data>
```

The above XML requests a label preview **(PREVIEW=True)** for the **`label.nlbl`** label **(FORMAT=label.nlbl)** in PDF format **(PREVIEW_FORMAT=PDF)**.

Structure of “Header” element

See the structure **/NICELAB/RQST_HEADER_OUT**.

FIELD NAME	FIELD DESCRIPTION
FORMAT	Contains the name of the label file together with the file extension (.NLBL). NOTE: To request a specific version of a label, use parameter ?v=<revnum> after the label name (where <revnum> is the revision number.
QUANTITY	Defines the number of labels to print.
QUANTITY_IDENTICALCOPIES	Defines the number of label duplicates to print. A default value is 1.
PRINTERNAME	Defines the name of the printer where labels print. This must be the exact printer name as installed on a Windows computer with Loftware Automation.
TRAY	Defines the name of a paper tray containing your media. Loftware Automation will use media from this paper tray for printing.
PRINTERSTATUS	Specifies whether to query the printer for its live status. This is a Boolean field.

JOBNAME	Defines the name for the job file in Windows Spooler. If omitted, the job name equals the name of the label file.
SPOOL_ID	Reserved for future use.
USER	Defines the user name. This is an optional field and is reserved for future use.
SYSTEM	Defines the system name. This is an optional field and is reserved for future use.
SYSTEM_TYPE	Specifies the system type. This is an optional field and is reserved for future use.
PRINT	Specifies whether to print the label. This is a Boolean field. NOTE: When providing value True, make sure that at least one <Data> element exists in the XML payload. In case of empty element, use the syntax <Data></Data> (not the shorten variant <Data />).
PREVIEW	Specifies whether to provide the label preview. This is a Boolean field. NOTE: When providing value True, make sure that at least one <Data> element exists in the XML payload. In case of empty element, use the syntax <Data></Data> (not the shorten variant <Data />).
PREVIEW_FORMAT	Defines the format of the label preview. The available formats are PDF, PNG, and JPEG. If you do not specify a format, PDF is provided.
EMAIL_RECIPIENTS	A comma-separated list of email addresses. When the label preview is generated, it will be sent to these recipients. You must configure and enable the email-sending action in Automation Configuration. A sample action is provided, but you must fill in the details for your infrastructure (e.g., SMTP server name, credentials).
PRINTERS	Specifies whether to provide the list of printer drivers that are available on the computer, where Loftware Automation runs. When the header also contains CLOUD_APIKEY and CLOUD_APIVERSION field, Automation will also return available cloud-connected printers. This is a Boolean field.
PRINT_JOB	Specifies whether to provide a binary print job back to SAP. The print job contains the selected label and the provided master data both converted to the label printer programming language. This is a Boolean field. NOTE: When providing value True, make sure that at least one <Data> element exists in the XML payload. In case of empty element, use the syntax <Data></Data> (not the shorten variant <Data />).
VARIABLES	Specifies whether to provide a list of variables and their properties from the selected label. This is a Boolean field.
CATALOG	Specifies whether to provide a list of all labels (Label Catalog) from DMS that are visible to Loftware Automation. This is a Boolean field.
CATALOG_ROOT	Defines the starting folder from which the label catalog creates. By default, all labels from the DMS will be included in the label catalog. Example value: `/Labels/DispatchLabels`

DATAMODEL	Specifies whether to generate a file with a data model definition. This is a Boolean field. All name-value pairs from XML payload are converted into Loftware-native .NLVR file that contains the definition of data sources (variables). You can use .NLVR file to import variables into the label template. A .NLVR file appears in the same folder where you have your Automation configuration (.MISX file).
CLOUD_PRINT	Specifies whether Automation should print a label directly to a printer or using a Cloud Print API (available for cloud-connected printers). You also have to provide a PRINT flag with value True. This is a Boolean field.
CLOUD_APIKEY	Defines the Cloud Print API subscription key. This is mapped from OCP-APIM-SUBSCRIPTION key from the /NICELAB/IF_CTRO configuration table.
CLOUD_APIVERSION	Defines the version of the Cloud Print API to use. This is mapped from API-VERSION key from the /NICELAB/IF_CTRL configuration table. [DEPRECATED – The Automation configuration will automatically use the appropriate version of the API]
CLOUD_LABELVERSION	Defines the version of the label to print. Default is no version and in that case, the latest approved version of the label template is used.
PRINTERSETTINGS	Defines custom printer settings that are used to overwrite currently applied printer settings. The printer settings are provided as a Base64-encoded representation of the Windows printer driver DEVMODE structure and must be compatible with the currently selected printer driver. For more information, see https://help.loftware.com/cloud/en/Content/Automation/UUID-1f792d48-23f2-0ab8-ea2f-8bb54e13163a.html#UUID-c39f8bc8-34af-a751-eb3a-eacb56c6486f and https://help.loftware.com/cloud/en/Content/Automation/UUID-b69d5885-4a40-dd10-e358-5d8db04b3725.html
LABELSETTINGS	Defines custom label settings that are used at print time to overwrite label settings defined in the label template (NLBL file), such as number of labels per page, label width, label height, offsets, label orientation etc. You must provide label settings in an XML message as described in the Loftware documentation. For more information, see: https://help.loftware.com/cloud/en/Content/Automation/UUID-1f792d48-23f2-0ab8-ea2f-8bb54e13163a.html#UUID-c39f8bc8-34af-a751-eb3a-eacb56c6486f
VARIANTS	Specifies whether Automation should create label variants. This is a Boolean field.
VARIANTS_UNLOCKEDVARIABLES	Defines the CSV list of variables (data sources) that must remain unlocked in the generated label variants. All variables not specified in this list will have their values hardcoded into the label template.
EXPORT_DATA	Specifies whether to run custom actions within the EXPORT_DATA placeholder in the Automation configuration. This is useful when you want to save the data provided in the XML payload to some

	custom location (e.g., database, files). Automation configuration provides an empty placeholder that you can configure yourself and adapt to your use case.
CUSTOM1 .. CUSTOM3	These are flags that you can use to control the execution of custom action workflows in the Automation configuration that you have configured yourself. This allows you to easily expand the Automation configuration for your requirements.

Structure of “Data” element

See the structure **/NICELAB/RQST_ITEM_DATA**.

For name-value pairs.

FIELD NAME	FIELD DESCRIPTION
Item	This is the XML element that contains name-value pairs. Value of “item” specifies “value” component in name-value pair.
Attribute “Id”	Specifies “name” component in the name-value pair.
Attribute “Type”	This attribute is used for providing report-formatted data. It will have no value when printing labels and will have the value “Report” when printing shipping documents.
Attribute “Position”	This attribute is used to identify the row number of the correspondent report item.

Structure of Automation’s feedback message

The result of data processing in Automation is provided back to ABAP Package in a synchronous response.

Successful execution

If there is no error while processing the data, ABAP Package receives an XML-formatted message. The XML message always contains all the elements, but some might be empty. For example, if you request a label preview, the **PrinterStatus**, **PrinterList,** and other elements remain empty.

Response structure:

```
<?xml version="1.0"?>
<Feedback>
    <ResponseText />
    <PrinterStatus />
    <Preview />
    <PrinterList />
    <LabelVariables />
    <PrintJob />
    <LabelCatalog />
    <Custom1 />
    <Custom2 />
    <Custom3 />
</Feedback>
```

Response parameters:

FIELD NAME	FIELD DESCRIPTION
ResponseText	Reserved for future use.
PrinterStatus	Contains live status as received from the label printer. Depending on the printer model and its state, the value might contain text descriptions “Out of labels”, “Out of ribbon”, “Not accessible”, and similar. The format and number of descriptions are printer-dependent.
Preview	Contains base64-encoded label preview in the selected file format.
PrinterList	Contains an XML list of printers visible to Loftware Automation. For the XML structure, refer to the topic **Structure of <PrinterList />** on page 97.
LabelVariables	Contains an XML list of variables from the selected label and their properties. For the XML structure, refer to the topic **Structure of <LabelVariables />** on page 98.
PrintJob	Contains base64-encoded binary content for the print job. Loftware Automation converts the selected label and the provided master data into the print job. The print job c ♣ ontains commands in the printer’s programming language, such as ZPL or SPL (Zebra or SATO programming language). Loftware Automation can create a print job for **any** printer if you have installed the Windows driver for it.
LabelCatalog	Contains an XML list of all labels from DMS that are accessible to Loftware Automation.
Custom1 .. Custom3	These fields contain the custom data that you want to send back into ABAP Package. These are optional fields.

Response type:

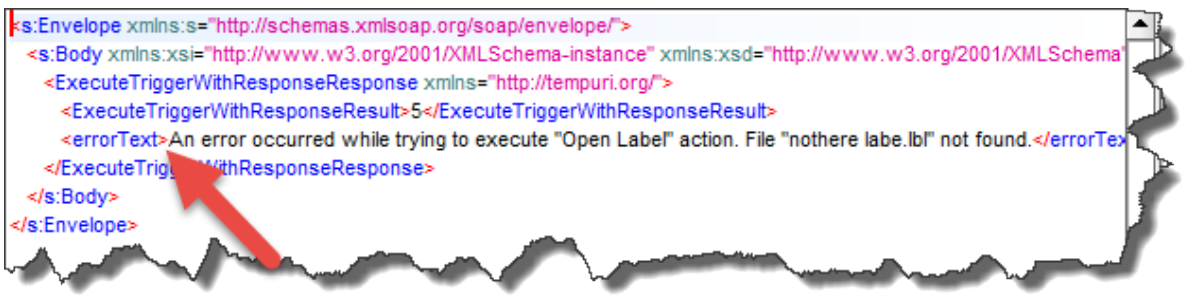
- RFC type G (HTTP trigger and Clout trigger) returns response in a HTTP body (as application/octet-stream content type).
- SOA (Web Service trigger) returns a response in the SOAP message. This response is base64-encoded.

Erroneous execution

If there is a processing error, such as “label not found”, “printer not available”, “wrong data for label objects” and others, the trigger in Loftware Automation raises an error and sends details back in the trigger response.

The structure of the error response depends on the trigger type:

- Web Service trigger provides the result in **`<errorText>`** element.



- HTTP trigger provides feedback in the HTTP header and the body of the message.

Structure of <PrinterList />

See the structure **/NICELAB/IF_PRINTER_LIST**.
There is a **Printer/Name** element for each printer.

FIELD NAME	FIELD DESCRIPTION
Name	The Name field can contain any of the following: • Name of the Windows printer driver as visible to Loftware Automation. • Name of the cloud-connected printer as registered in Loftware Control Center. You need a Loftware Cloud subscription for this functionality.
CloudPrinter	Specifies whether the current printer is a cloud-connected printer or printer visible through Loftware Automation. This is a Boolean field.
Location	Provides the printer location. This information is read from the Windows printer driver “Location” field.

Example:

```
<Printer>
    <Name>SATO CL4NX 203dpi</Name>
    <CloudPrinter>False</CloudPrinter>
</Printer>
<Printer>
    <Name>ZEBRA R-402</Name>
    <CloudPrinter>False</CloudPrinter>
    <Location>Room 212</Location>
</Printer>
<Printer>
    <Name>ZEBRA ZD420-300dpi ZPL</Name>
    <CloudPrinter>True</CloudPrinter>
```

```
    <Location>Site A</Location>
</Printer>
```

Structure of <LabelVariables />

See the structure **/NICELAB/IF_LABEL_VARIABLE**.
Also, see the topic **Get Label Information** in the Automation user guide.
There is a **Variable** element for each variable in the label.

FIELD NAME	FIELD DESCRIPTION
Name	Contains variable name.
Description	Contains variable description.
DefaultValue	Contains default value as defined for the variable during the label design process.
Format	Contains the acceptable type of variable content (characters).
IsPrompted	Contains information whether the variable is prompted at a print time or not.
PromptText	Contains text that prompts the user to input the value.
CurrentValue	Contains the actual value that is used for printing.
IncrementType	Contains information whether the variable is defined as a counter or not. If identified as a counter, it tells what kind of counter it is.
IncrementStep	Contains information about the counter step. Counter value increments/decrements for this value on the next label.
IncrementCount	Contains information about the point of counter value incrementing/decrementing. Usually, the counter changes value on every label, but that can be changed.
Length	Contains a maximum number of stored characters in a variable.
IsPickListEnabled	Contains information on whether the user selects variable values from a picklist.
PickListValues	Contains the actual (selectable) picklist values.

Example:

```
<Variable>
    <Name>FIELD1</Name>
    <Description />
    <DefaultValue>Loftware</DefaultValue>
    <Format>All</Format>
    <IsPrompted>True</IsPrompted>
    <PromptText />
    <CurrentValue>Loftware</CurrentValue>
    <IncrementType>None</IncrementType>
    <IncrementStep>0</IncrementStep>
    <IncrementCount>0</IncrementCount>
    <Length>20</Length>
    <IsPickListEnabled>False</IsPickListEnabled>
    <PickListValues>
```

```
        <PickListValue>1</PickListValue>
        <PickListValue>2</PickListValue>
        <PickListValue>3</PickListValue>
    </PickListValues>
</Variable>
```

Structure of <LabelCatalog />

See the structure **/NICELAB/IF_LABEL_LIST**.
There is a **Labels/Label** element for each label in the DMS.

FIELD NAME	FIELD DESCRIPTION
Name	Name of the label including the path from the document storage root. The server name and port number are not included, Loftware Automation knows the location of its DMS.
Width	Contains the label width in 1/1000 of a millimeter.
Height	Contains the label height in 1/1000 of a millimeter.
PrinterName	Contains printer name for which the label has been designed. NOTE: You can specify another printer for printing. Loftware Automation adapts the label template for the new printer type.
Description	Contains the description as configured in the label.
Revisions	Contains the version of the last accessible revision of the label template. NOTE: Label visibility is determined by the role-based access control (RBAC). When Loftware Automation runs as a member of the Operator profile (best practice), it can only access the approved label revisions.

Example:

```
<Labels>
    <Label>
        <Name>/Demo/SAP ABAP Package/label.nlbl</Name>
        <Width>100000</Width>
        <Height>75000</Height>
        <PrinterName>CAB A3 203DPI</PrinterName>
        <Description></Description>
        <Revisions>4</Revisions>
    </Label>
…
</Labels>
```

API reference

/NICELAB/ namespace

All SAP objects that enable communication with the Loftware Automation are developed inside the /NICELAB/ namespace in package/development class /NICELAB/INTERFACE. This exclusive namespace ensures code independence from the SAP core system or any custom development projects in your SAP deployment that you might develop in the customer namespaces “Z” or “Y”.

The names of the Loftware custom-developed SAP objects do not get into conflict with the names of standard SAP objects. The Loftware objects are also not subject to change if a patch is applied, or if an upgrade is undertaken. Using the unique namespace not only prevents naming conflicts but also prevents unauthorized modification of distributed objects that were built in the unique /NICELAB/ namespace.

ABAP class (API) methods

Loftware provides a standard interface class (API) for SAP, which enables the communication between SAP ERP systems and Loftware Automation Enterprise.

API class /NICELAB/CL_INTERFACE_ROOT

API class – **/NICELAB/CL_INTERFACE_ROOT** – is an abstract class and, as such, cannot be used directly in the customer application. To use the API in the customer application, a new class with /NICELAB/CL_INTERFACE_ROOT as parent (super) class must be defined in the customer system. An example child class /NICELAB/CL_INTERFACE_DEMO is also available as a part of the ABAP Package.

The next section describes public API class methods in detail:

/NICELAB/CL_INTERFACE_ROOT->CONSTRUCTOR

Importing Parameters:
I_OPERATION_MODE (TYPE /NICELAB/INTERFACE_OP_MODE): Interface operation mode – Web Service or HTTP request (SOA/HTTP)

Description:
This is a constructor method of the API interface. API operation mode (SOA/HTTP) can be explicitly set for the API instance via *I_OPERATION_MODE* importing parameter. If this is not provided, the default operation mode is read interface configuration table.

/NICELAB/CL_INTERFACE_ROOT ->LOGICAL_PORT_SET

Importing Parameters:
I_LP (TYPE PRX_LOGICAL_PORT_NAME): Logical port

Description:
This method can be used to override the default logical port configured for the interface in the /NICELAB/V_IF_CT configuration view. The method is only valid for Web Service operation mode (“SOA”). This can be used in the calling application, to set the desired Loftware Automation target system.

/NICELAB/CL_INTERFACE_ROOT ->RFC_DESTINATION_SET

Importing Parameters:
I_RFCDEST (TYPE RFCDEST): RFC destination of type G (External HTTP connection)

Description:
This method can be used to override the HTTP connection configured for the interface in the /NICELAB/V_IF_CT configuration view. The method is only valid for HTTP operation mode ("HTTP"). This can be used in the calling application to set the desired Loftware Automation target system.

/NICELAB/CL_INTERFACE_ROOT->DO_PRINT_XML

Importing Parameters:
I_NL_DATA (TYPE /NICELAB/RQST_DATA_OUT): Request Data
I_WITH_RESPONSE (TYPE BOOLEAN DEFAULT ABAP_TRUE): Get Response
I_WRITE_SPOOL (TYPE BOOLEAN OPTIONAL): Write Request/Response To Spool
I_SPOOL_TITLE_ADD (type STRING DEFAULT 'Request'): Spool Title Addition
I_CLOUD_TRIGGER (BOOLEAN, optional, DEFAULT ABAP_TRUE): Generate Cloud Trigger Request

Description:
This is the main method to send the print request data to Loftware Automation. Print request data contains print meta-data information, such as preview type (PDF, JPG, etc.), request printer status attribute, number of print copies, and the variable data to be printed. Print data is provided as name-value pairs which are defined based on the printing application.

Depending on the interface operation mode – "SOA" (Web service) or "HTTP" (RFC type G) – the logical port or RFC connection must be specified.

- **For the "SOA" operation.** If no port is specified, the default port from the configuration is used. If no default PORT is found in /NICELAB/V_IF_CT, the port marked as default for /NICELAB/CO_NL_AUTOMATION proxy class is used (defined and marked as default in SOAMANAGER transaction).
- **For the "HTTP" operation.** If no RFC is explicitly specified, the default RFC from the configuration is used. If no default RFC is found in /NICELAB/V_IF_CT, the call is aborted, and an exception is raised.

When using the I_WITH_RESPONSE parameter, the application can control if the response is returned for the print request (this only applies for Web Service operation mode).
With parameter I_WRITE_SPOOL, the application can override the setting for logging the print request/response data to the SAP spooler. If this parameter is not set, then the default setting for spool generation is taken from the /NICELAB/V_IF_CT configuration view.

Method parameter I_SPOOL_TITLE_ADD is only used if Spool logging is enabled (see parameter I_WRITE_SPOOL). The string specified here is added to the default spool title, depending on the spool content (for example to add object key e.g., order number to the spool title, for easier monitoring).
Parameter I_CLOUD_TRIGGER is used to generate an HTTP request to a Loftware Cloud Trigger REST service. When this parameter is set, additional HTTP headers are added to the HTTP request (API-VERSION, OCP-APIM-SUBSCRIPTION-KEY). This parameter is not used with SOA requests.

/NICELAB/CL_INTERFACE_ROOT->RESPONSE_ERROR_GET

Importing Parameters:
I_DIALOG (TYPE ABAP_BOOL): Show response error in a dialog
Returning Parameters:
E_ERROR_DESC (TYPE STRING): Error description (as string)

Description:
This method returns a description of the error, which might happen during the Loftware Automation print process in the *E_ERROR_DESC* variable. If the *I_DIALOG* parameter is set, the error message is displayed in a popup dialog.

/NICELAB/CL_INTERFACE_ROOT=>REQUEST_PREVIEW_XML

Importing Parameters:
I_NL_DATA (TYPE /NICELAB/RQST_DATA_OUT): Request Data

Description:
This method generates and displays XML request that is sent to Loftware Automation. This can be used for evaluation purposes (for example, if the application data was correctly mapped to XML structure).

/NICELAB/CL_INTERFACE_ROOT=>GET_API_INSTANCE

Importing Parameters:
I_API (TYPE /NICELAB/IF_API): Loftware Interface API Type
IS_IF_OPERATION (TYPE /NICELAB/IF_OPERATION): Loftware Interface: Operation Mode
Returning Parameters:
EO_NL_API (TYPE REF TO /NICELAB/CL_INTERFACE_ROOT): Loftware Automation Interface (API) reference

Description:
This static method returns an instance of a requested API class type in specified operation mode. Different API class types can be instantiated:

- LABEL – returns a reference to Loftware LABEL API Interface
- PRINTER - returns a reference to Loftware PRINTER API Interface
- DEMO - returns a reference to Loftware DEMO API Interface

/NICELAB/CL_INTERFACE_ROOT->GET_RESPONSE_TYPE

Importing Parameters:
I_FORMAT (TYPE /NICELAB/RESPONSE_FORMAT): Response Format
Returning Parameters:
RV_BIN_DATA (TYPE XSTRING): Loftware Interface: Operation Mode

Description:
This method returns the requested response type (PDF, JPG, or PNG) in binary format from the print request response.

Class /NICELAB/CL_INTERFACE_DEMO

Part of the API interface package is also an example successor class of the API /NICELAB/CL_INTERFACE_ROOT class. The example class is called /NICELAB/CL_INTERFACE_DEMO. For additional usability, the example extends the basic ABAP Package interface with additional methods:

/NICELAB/CL_INTERFACE_DEMO->PRINTER_STATUS_GET

Returning Parameters:
R_PRINTER_STATUS (TYPE STRING): Returns printer status in case of printer error.

Description:
If the printer status is requested in the original request to Loftware Automation, this method returns the printer status description in case of errors.

/NICELAB/CL_INTERFACE_DEMO->RESPONSE_DISPLAY

Importing Parameters:
I_FORMAT (TYPE /NICELAB/RESPONSE_FORMAT): Response format (PDF, JPG, etc.)

Description:
If a print preview is specified in the request to Loftware Automation, this method can be used to read the response and to present the result of a requested print. Different formats can be specified. The demo class is implementing support for PDF response types.

/NICELAB/CL_INTERFACE_DEMO->PRINT_INFORMATION_GET

Returning Parameters:
R_PRINT_INFO (TYPE STRING): Returns print job information

Description:
Loftware Automation can provide additional information about the submitted print job. This method can be used to read such information from the response.

Class /NICELAB/CL_IF_CONFIG

This class provides the interface to the Loftware interface configuration settings and can also be reused in customer developments to read Loftware interface-specific settings.

/NICELAB/CL_IF_CONFIG=>CLASS_CONSTRUCTOR

Parameters:
N/A

Description:
Buffers the Loftware API Interface configuration table(s)

/NICELAB/CL_IF_CONFIG=>GET_LOGICAL_PORT

Returning Parameters:
R_VALUE (TYPE /NICELAB/IF_CONFIG_VALUE): Loftware Interface: Configuration Value

Description:
Returns logical port setting from the configuration (configuration key = 'SOA_LP')

/NICELAB/CL_IF_CONFIG=>GET_RFC_DESTINATION

Returning Parameters:
R_VALUE (TYPE /NICELAB/IF_CONFIG_VALUE): Loftware Interface: Configuration Value

Description:
Returns RFC destination setting from the configuration (configuration key = 'HTTP_RFC')

/NICELAB/CL_IF_CONFIG=>GET_PRINTER_STATUS

Returning Parameters:
R_VALUE (TYPE /NICELAB/IF_CONFIG_VALUE): Loftware Interface: Configuration Value

Description:
Returns printer status (pooling True/False) from the configuration (configuration key = 'PRINTER_STATUS')

/NICELAB/CL_IF_CONFIG=>GET_OPERATION_MODE

Returning Parameters:
R_VALUE (TYPE /NICELAB/IF_CONFIG_VALUE): Loftware Interface: Configuration Value

Description:
Returns default (system wide) API operation mode from the configuration (configuration key = 'OPERATION_MODE')

/NICELAB/CL_IF_CONFIG=>GET_CONFIG_ID_VALUE

Importing Parameters:
I_CONFIG_ID (TYPE /NICELAB/IF_CONFIG_ID): Loftware Interface: Configuration ID
ReturningParameters:
R_VALUE (TYPE /NICELAB/IF_CONFIG_VALUE): Loftware Interface: Configuration Value

Description:
Returns configuration key value for requested configuration ID *(I_CONFIG_ID)*

/NICELAB/CL_IF_CONFIG=>GET_DATA_MODEL

Returning Parameters:
R_VALUE (TYPE /NICELAB/IF_CONFIG_VALUE): Loftware Interface: Configuration Value

Description:
Returns (system-wide) print request header setting to generate a variable data model from the print request (configuration key = 'DATA_MODEL')

Class /NICELAB/CL_IF_UTIL

This class interface provides Loftware global constants and some common methods for string processing, error display, etc.

Constants:
CO_RESP_FORMAT_PDF (*type* /NICELAB/RESPONSE_FORMAT) *value* 'PDF'
CO_RESP_FORMAT_PNG (*type* /NICELAB/RESPONSE_FORMAT) *value* 'PNG'
CO_RESP_FORMAT_JPG (*type* /NICELAB/RESPONSE_FORMAT) *value* 'JPG'
CO_OP_MODE_SOA (*type* /NICELAB/INTERFACE_OP_MODE) *value* 'SOA'
CO_OP_MODE_HTTP (*type* /NICELAB/INTERFACE_OP_MODE) *value* 'HTTP'
CO_XML_TRUE (type STRING) *value* 'TRUE'
CO_XML_FALSE (type STRING) *value* 'FALSE'
CO_MEM_ID_OP_MODE Memory ID for API operation mode
CO_API_LABEL (*type* /NICELAB/IF_API) *value* 'LABEL'
CO_API_DEMO (*type* /NICELAB/IF_API) *value* 'DEMO'
CO_API_PRINTER (*type* /NICELAB/IF_API) *value* 'PRINTER'
CO_OUTPUT_DEV_BIN (type RSPOPNAME) *value* ''NLBN' (Binary spool output device)
CO_OUTPUT_DEV_PDF (type RSPOPNAME) *value* 'NLRS' (PDF spool output device)
CO_MSGID_NICELABEL_IF (type MSGID) *value* ''/NICELAB/INTERFACE'
CO_ITEMID_VARIANT_FILENAME (*type* STRING) *value* '_VARIANT_FILENAME' (XML item ItemID constant in case Create Variant option is activated)

/NICELAB/CL_IF_CONFIG=>STRING_TO_TAB

Importing Parameters:
IV_STRING (*type* STRING)
Returning Parameters:
ET_CTAB (*type S*TANDARD TABLE)

Description:
Convert strings to the table

/NICELAB/CL_IF_CONFIG=>BOOL_2_TF

Importing Parameters:
I_BOOL (*type* BOOLEAN)
Returning Parameters:
R_TRUEFALSE (*type* STRING)

Description:
Convert ABAP Boolean value ('X', '') to XML True/False

/NICELAB/CL_IF_CONFIG=>TF_2_BOOL

Importing Parameters:
I_TRUEFALSE (*type* STRING)
Returning Parameters:
R_BOOL (*type* BOOLEAN)

Description:
Converts XML Boolean value ('True', 'False') to ABAP Boolean value ('', 'X')

/NICELAB/CL_IF_CONFIG=>CONV_STRING_2_ERROR_TAB

Importing Parameters:
I_TITLE (*type* /NICELAB/IF_S_NOTE-LINE *optional*)
I_ERROR (*type* STRING)
Returning Parameters:
RT_ERR_TAB (*type* /NICELAB/IF_NOTE_TAB)

Description:
Converts string to error table

/NICELAB/CL_IF_CONFIG=>MESSAGE_POPUP

Importing Parameters:
I_TITLE (*type* SYTITLE *optional*)
I_MESSAGE (*type* STRING)

Description:
Display message string (*I_MESSAGE)* in a popup dialog

/NICELAB/CL_IF_CONFIG=>GET_PROCESS_OBJ_CHAR

Importing Parameters:
I_OBJNUM (*type* OBJNUM): Object number
I_PROCESS (*type* /NICELAB/PROCESS *optional* SY-TCODE): Integrated Process (e.g., transaction code)
I_KEY01 (*type* /NICELAB/IF_PROC_KEY01 *optional*): Generic Key 1
I_KEY02 (*type* /NICELAB/IF_PROC_KEY02 *optional*): Generic Key 2
I_SPRAS (*type* SPRAS *optional):* Language
Returning Parameters:
RT_OBJ_CHAR (*type* /NICELAB/PROC_OBJ_CLAS_T) : Material characteristics

Description:
Reads the Loftware integrated process material characteristics configuration table **/NICELAB/PROC_MC** and returns characteristics for all configured class / characteristic combinations.

/NICELAB/CL_IF_CONFIG=>GET_PROCESS_CONFIG

Importing Parameters:
I_PROCESS (*type* /NICELAB/PROCESS *optional* SY-TCODE): Integrated Process (e.g., transaction code)
I_KEY01 (*type* /NICELAB/IF_PROC_KEY01 *optional*): Generic Key 1
I_KEY02 (*type* /NICELAB/IF_PROC_KEY02 *optional*): Generic Key 2
I_USERNAME (*type* UNAME *optional* SY-UNAME*):* Username
Returning Parameters:
R_IF_PROC (*type* /NICELAB/IF_PROC): Loftware Interface: Process Settings

Description:

This method can be used to read the Loftware integrated process configuration from the /NICELAB/IF_PROC table. The method also considers generic entries in generic key fields and the username field, meaning that the pattern entry using the '*' character can be used for creating more generic entries.

Class /NICELAB/CL_IF_PRINTER

This API class provides methods for retrieving printer information from Loftware Automation, for example, reading connected printers (local and Cloud) and refreshing the Loftware printer list in SAP.

/NICELAB/CL_IF_PRINTER=> GET_LIST

Returning Parameters:
E_ERROR (*type* STRING): Error description
ET_PRINTER_LIST (*type* /NICELAB/IF_PRINTER_LIST_TAB): Printer List

Description:
This method sends a request to Loftware automation with header element *<PRINTER>True</PRINTER>* . Such a request will return a list of all printers connected to Loftware Automation. If configuration IDs OCP-APIM-SUBSCRIPTION-KEY and API-VERSION are configured, additional HTTP headers are sent to Automation, and a list of returned printers will also contain Cloud-enabled printers.

/NICELAB/CL_IF_PRINTER=> REFRESH_PRINTER_LIST_SAP

Importing Parameters:
I_COMMIT (*type* BOOLEAN): Commit changes to DB

Description:
This method is used for refreshing a list of Loftware printers in the SAP system. List of printers is received from Loftware Automation (see /NICELAB/CL_IF_PRINTER=>GET_LIST method) and stored in an SAP DB table /NICELAB/IF_PRNT.

/NICELAB/CL_IF_PRINTER=> IS_CLOUD_PRINTER

Importing Parameters:
I_PRINTER_NAME (*type* STRING): Printer Name
Returning Parameters:
R_CLOUD (*type* BOOLEAN): Cloud Printer True/False

Description:
This method checks if the specified Loftware printer is a Cloud printer. This information is stored locally, in the SAP system, in table /NICELAB/IF_PRNT (attribute CLOUD_PRINTER).

Class /NICELAB/CL_IF_LABEL

This API class provides methods for retrieving label information from Loftware automation.

/NICELAB/CL_IF_LABEL=> GET_LIST

Returning Parameters:
E_ERROR (*type* STRING): Error description
ET_LABEL_LIST (*type* /NICELAB/IF_LABEL_LIST_TAB): Label List

Description:
This method sends a request to Loftware automation with header element *<CATALOG>True</CATALOG>*. Such a request will return a list of all labels from the Loftware Automation repository.

/NICELAB/CL_IF_LABEL=> GET_LABEL_VARIABLES

Returning Parameters:
I_LABEL (*type* /NICELAB/LABEL): Label Name
I_DIALOG (*type* BOOLEAN): Execution in dialog
Returning Parameters:
ES_LABEL_VAR (*type /NICELAB/IF_LABEL_VAR*): Label Variables
E_ERROR (*type* STRING): Error description

Description:
This method sends a request to Loftware automation with header element *<VARIABLES>True</VARIABLES>*. Such request will return a list of variables used on a specified label in exporting parameter ES_LABEL_VAR. The error description is returned in the E_ERROR variable in case of errors. If the I_DIALOG parameter is set to *True ('X')* the error message will be displayed in a popup dialog.

Support

Online help

You can find the latest builds, updates, workarounds for problems, and Frequently Asked Questions (FAQ) on the product website a www.loftware.com.

For more information, please refer to:

- User guides: https://help.loftware.com/cloud/en/Content/LoftwareCloud.htm
- Loftware Support: https://www.loftware.com/customer-center/technical-support/submit-a-support-ticket
- Loftware trainings and tutorials: https://www.loftware.com/customer-center/training-and-tutorials

NOTE: If you have a Service Maintenance Agreement (SMA), please contact the premium support as specified in the agreement.

Troubleshooting

Error creating logical port in transaction SOAMANAGER

When creating a logical port, you might receive the following error:
SRT Framework exception: Error in WSDL parsing: There was an exception in Library Handler

Error cause: SAP connected to the provided address where Web Service trigger is active in Loftware Automation. However, the trigger response was not formatted in the way the ABAP Package expects it to be.

The WSDL style in the trigger is not configured to “document”.
The reason for receiving this error message is that you are probably not running the Automation triggers provided with ABAP Package. You have configured your triggers instead.

To resolve the issue:

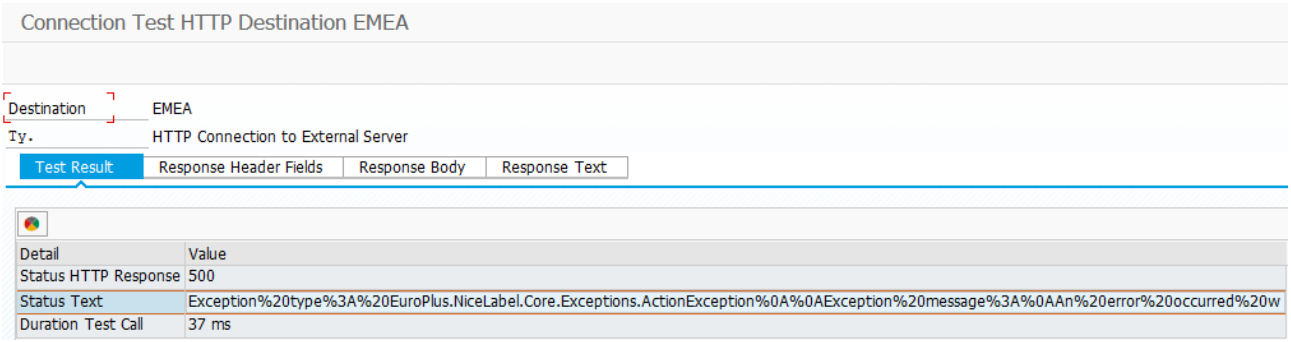
1. Open the trigger configuration in **Loftware Automation Builder**.
2. Edit your Web Service trigger.
3. In the **Settings** tab, select **Document** for WSDL Style.
4. Save the trigger.
5. Deploy the new configuration.

Problem executing “Connection test” in SM59 transaction

SM59 transaction helps you configure the “HTTP connection to External Server”. This single configuration step establishes communication between ABAP Package and Loftware Automation over the HTTP interface. The result is the “RFC destination” to which ABAP Package forwards the requests from your application.

The SM59 transaction includes the “Connect Test” button, which sends data to the RFC destination. When you execute a connection test, you receive an error response. You receive the 500 HTTP status response and

a status message containing the “`Exception type: EuroPlus.NiceLabel.Core.Exceptions.ActionException…`”



Despite the error report, the test was executed successfully. You have received a response from Loftware Automation, which means you have a clear communication path with Loftware Automation.

What you see is a standard response from Loftware Automation for a detected error. The “Connection Test” button did not send the data formatted in an XML structure as Loftware Automation expected. Therefore, the response is an error message.

To summarize, the error that you see after running the “Connection Test” proves that you can communicate with Loftware Automation. To test your ABAP Package, run the provided demo program /NICELAB/INTERFACE_DEMO. This program sends the XML data the way Loftware Automation expects it to be.

Failing to create Label Catalog

When you set value of **<CATALOG />** XML element to **True** (and optionally provide the starting folder with **<CATALOG_ROOT />** XML element), the label catalog does not create. An empty value or previous version of the label catalog is returned.

NOTE: Label catalog creates with labels from the DMS, not from any other source location.

Label catalog is not created in the main trigger that receives data from SAP. Creating the label catalog might take a while, so the process is asynchronous. The main trigger does not wait for a response. The label catalog creation is offloaded to another trigger within the Loftware Automation configuration with the name **Label Catalog Async Updater**. When the catalog is created, it is saved as an XML file in the folder with Automation configuration.

Verify the following:

1. Is this the first time you have requested the label catalog? If yes, the empty catalog returns. When the label catalog is created and saved to XML, the next request provides it.
2. If you have received the old label catalog (there are no new labels included in the catalog), the creation process might not have been completed already. In Loftware Manager, see the live status of the trigger **Label Catalog Async Updater**. If you see the progress indicator, the trigger is still busy creating the catalog.
3. The previous label catalog generation might have failed and did not clear the signal file that prevents the concurrent generation of catalog on several Automation servers.

   Do the following:

a. Make sure the **Label Catalog Async Updater** trigger is not currently in use (there must be no progress indicator next to the trigger).
b. Do this check on each of your deployed Automation servers.
c. In **Control Center**, go to the **Documents** tab.
d. Navigate to the folder, where you saved the Automation configuration.
e. Delete the file **LabelCatalogUpdateInProgress.txt**.

Repairing ABAP Package enhancements after SAP upgrade

Customers continuously upgrade their SAP to newer versions/feature packs to benefit from feature enhancements and bug fixes. During the upgrade, existing SAP standard objects are overwritten. Customers can retain the modified object as in older systems or adapt to new changes through transactions SPAU and SPAU_ENH.

ABAP Package contains the enhancement spot that hooks into the standard outbound delivery transaction and invokes label printing through Loftware. All ABAP Package items are contained within /NICELAB/ namespace.

When running tcode SPAU_ENH after the SAP upgrade, it might display a red traffic light (manual adjustment) next to the following items:
/NICELAB/ENH_IF_RVADDN01
/NICELAB/ENH_IF_RLE_DELNOTE

Customers cannot adjust these enhancements due to the missing change license for /NICELAB/ namespace. You might see the following errors:

- Namespace /NICELAB/ does not exist
- No valid change license exists for namespace /NICELAB/

These custom objects are no longer working and need an adjustment.

Do the following:

1. You must add the /NICELAB/ namespace in your system with a namespace repair key.
2. The repair key to use is **28595060430262820612**.
3. The procedure to release a namespace for repairs is documented here:
    a. If you see the error “No valid change license exists for a namespace /NICELAB/”: https://help.sap.com/doc/saphelp_nw74/7.4.16/en-us/bd/ddbe0bac5c11d2850e0000e8a57770/content.htm?no_cache=true
    b. If you see the error “Namespace /NICELAB/ does not exist”: https://help.sap.com/saphelp_snc700_ehp01/helpdata/en/bd/ddbe08ac5c11d2850e0000e8a57770/content.htm?no_cache=true
    Make sure to use “C - Recipient” for the namespace role.

Then the enhancement point adjustments should work.