Wojciech Eichert, Krzysztof Łuka, Mateusz Nowak

Interface Monitoring and Error Handling with SAP® AIF

Rheinwerk Publishing

What You'll Learn

In this E-Bite, we'll walk through how to use SAP Application Interface Framework to monitor SAP interfaces. We start with an overview of SAP Application Interface Framework, and then we'll dive into monitoring and error handling with Transaction /AIF/ERR. After that, we'll explore the utility of Transaction /AIF/IFMON, alerts management, statistical reports, and built-in authorizations. Finally, we'll look at SAP Process Observer and SAP Solution Manager integration.

1 Getting Started

There are various technologies via which you can exchange data between your SAP application system and the rest of the world. You can use RFC and call a remote function in an external system or let the external system call a function in the SAP system. You can use a file as an exchange medium between peers. The SOAP protocol is available in two different variants: XI 3.0 or the wider standard SOAP 1.1. You can also make or accept OData calls. This list of possibilities is not complete, of course. With the numerous options you have, it is a challenge to have control over all those interfaces, in terms of both their consistent development and their monitoring.

SAP Application Interface Framework may be an answer to those challenges. SAP Application Interface Framework is an add-on that you can install in any SAP application system. In SAP S/4HANA, it is even easier because it is already one of the standard components.

Pricing and Licensing of SAP Application Interface Framework

Although SAP Application Interface Framework is part of the standard SAP S/4HANA delivery, it has a separate license. It is similar when you install it as an add-on. For more details on pricing and licensing rules, visit https://www.sap.com/products/application-interface-mgmt.html or contact SAP directly.

As an interface framework, SAP Application Interface Framework has two main pillars: design and monitoring. Using its rich functionalities, you can design interfaces in a very consistent way, utilizing the mapping engine with checks, structure and value mappings, and actions. The monitoring

features, in turn, offer the ability to monitor all your SAP application interfaces, regardless of their technology, in a single monitoring transaction. This means that with SAP Application Interface Framework, you can forget about other monitors like SXI_MONITOR for SOAP, WE02/BD87 for IDocs, or even the custom ones you developed to monitor file interfaces, and instead benefit from the single SAP Application Interface Framework monitor.

It is not obligatory to use both pillars every time. There are three ways to use SAP Application Interface Framework:

1. **Design and monitor**
   In this scenario, you design your interfaces in SAP Application Interface Framework, using its mapping engine, and then you process interface messages in SAP Application Interface Framework runtime and monitor them with this tool as well.
2. **Monitor only**
   Here you only monitor the interfaces with SAP Application Interface Framework, without processing them in the SAP Application Interface Framework runtime.
3. **Design only**
   Here you design your interfaces in SAP Application Interface Framework and process the messages in SAP Application Interface Framework runtime, but you do not use the monitor.

Processing in SAP Application Interface Framework Runtime vs. Monitoring Only

If you want to use the SAP Application Interface Framework mapping engine to validate a message, convert it structurally, or map its values, you need to process an interface message in SAP Application Interface Framework runtime. This means that a message will be passed to SAP Application Interface Framework, where it will undergo any mapping or validation steps that you designed. At the end, an action (also of your design) needs to be executed, which will use the converted message to either create the business document in the system or, for outbound interfaces, send it to the external system.

> If you are focused on the monitoring features of SAP Application Interface Framework only, you can use the enablement mode. In this mode, a message is not processed in SAP Application Interface Framework runtime, which means that you are not changing any processing logic. SAP Application Interface Framework's role is only to monitor a message and provide you with extensive information on the status of each message, along with options for resolving the error.

It is always a challenge to implement SAP Application Interface Framework in an organization with robust and mature SAP implementation. Fear of changing a complex interface design might be a showstopper. But because you are not obliged to use the full scope of SAP Application Interface Framework from the beginning, you can implement it gradually in your organization. Best practice is to start with a rather simple monitoring enablement with SAP Application Interface Framework to become familiar with the tool before using all the monitoring features it provides.

That approach has the following advantages:

- **Transparency**
  Having all interface messages in one place makes it easier to identify and react to issues.
- **Consistency**
  The monitoring process for each interface is very similar.
- **Gentle learning curve**
  Monitoring users need to learn only one tool.

In this E-Bite, we will focus on the monitoring capabilities of SAP Application Interface Framework. We will not be covering any of the interface design topics. We will guide you through various monitoring functionalities that SAP Application Interface Framework offers, and we will explain how to implement them. We prepared the examples based on a single IDoc interface so that you can easily follow it in your system. Because the monitoring features of SAP Application Interface Framework are, in most cases, technology-independent, the examples provided in this E-Bite can be

applied to your other interfaces as well. We hope you enjoy reading this E-Bite!

To monitor your IDocs using SAP Application Interface Framework, you have to create new interfaces. All SAP Application Interface Framework interfaces belong to a specific namespace, so we have to create our own namespace for the purposes of this E-Bite.

To create a new namespace, go to Transaction /AIF/CUST and open the **Interface Development • Define Namespace** node. In the first pop-up window, provide your namespace name and description, as shown in Figure 1.

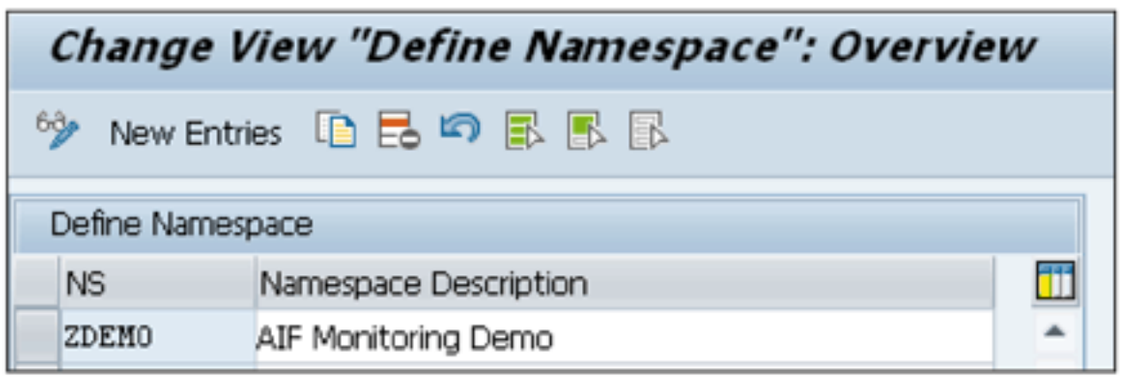

Figure 1 SAP Application Interface Framework Namespace Creation

In our scenario, we will monitor IDocs; therefore, we can use a helpful transaction that will help us define an interface and create ABAP structures necessary for monitoring IDocs in SAP Application Interface Framework.

Go to Transaction /AIF/IDOC_GEN and fill the required fields.

Because we are interested in monitoring the ORDERS.ORDERS05 IDoc, we have to use exactly these values in the **Basic Type** and **Message Type** fields.

We recommend using the **Prefix Structure** field to define how the generated structures will start. Thanks to this, we can stick to the guidelines of the system you are working on.

In the **IDoc Data Structure** field, provide the name that will be used as a structure name in the generated IDoc interface. For the namespace, enter the "ZDEMO" namespace created in the previous step. Type "IIDSORDE01" as the interface name, "1" for the interface version, and provide a description

of the interface. These values will be used to create a new IDoc interface. You should provide the information as shown in Figure 2.

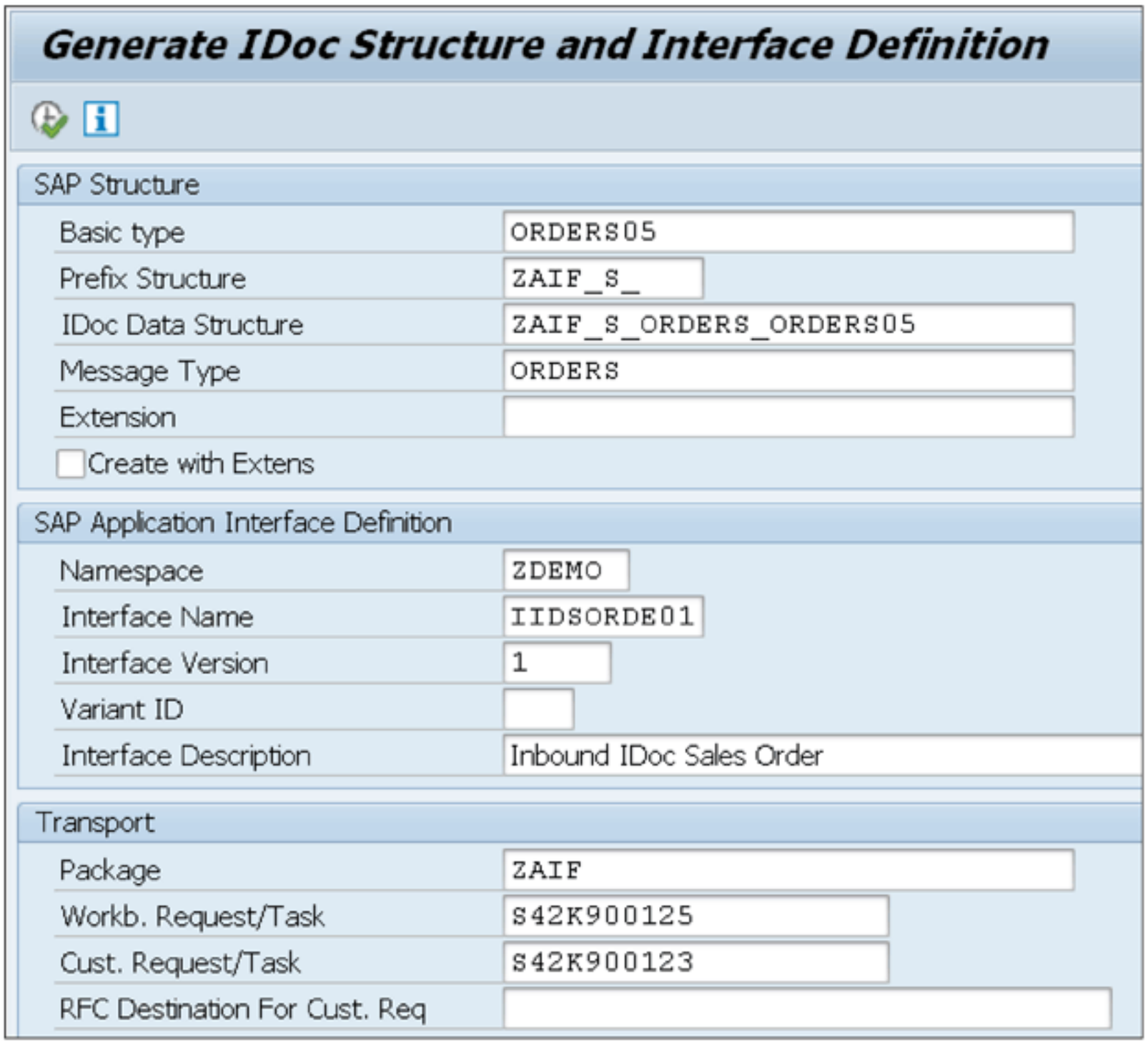

Figure 2 Generation of ABAP Structure for ORDERS.ORDERS05 IDoc

Click the **Execute** button or press F8. A new log window will appear with information about the generated structures and interface, as shown in Figure 3.

Now you can preview the created interface. Go to Transaction /AIF/CUST and then open **Interface Development • Define Interface**. Type "ZDEMO" in **Namespace** field, as shown in Figure 4.

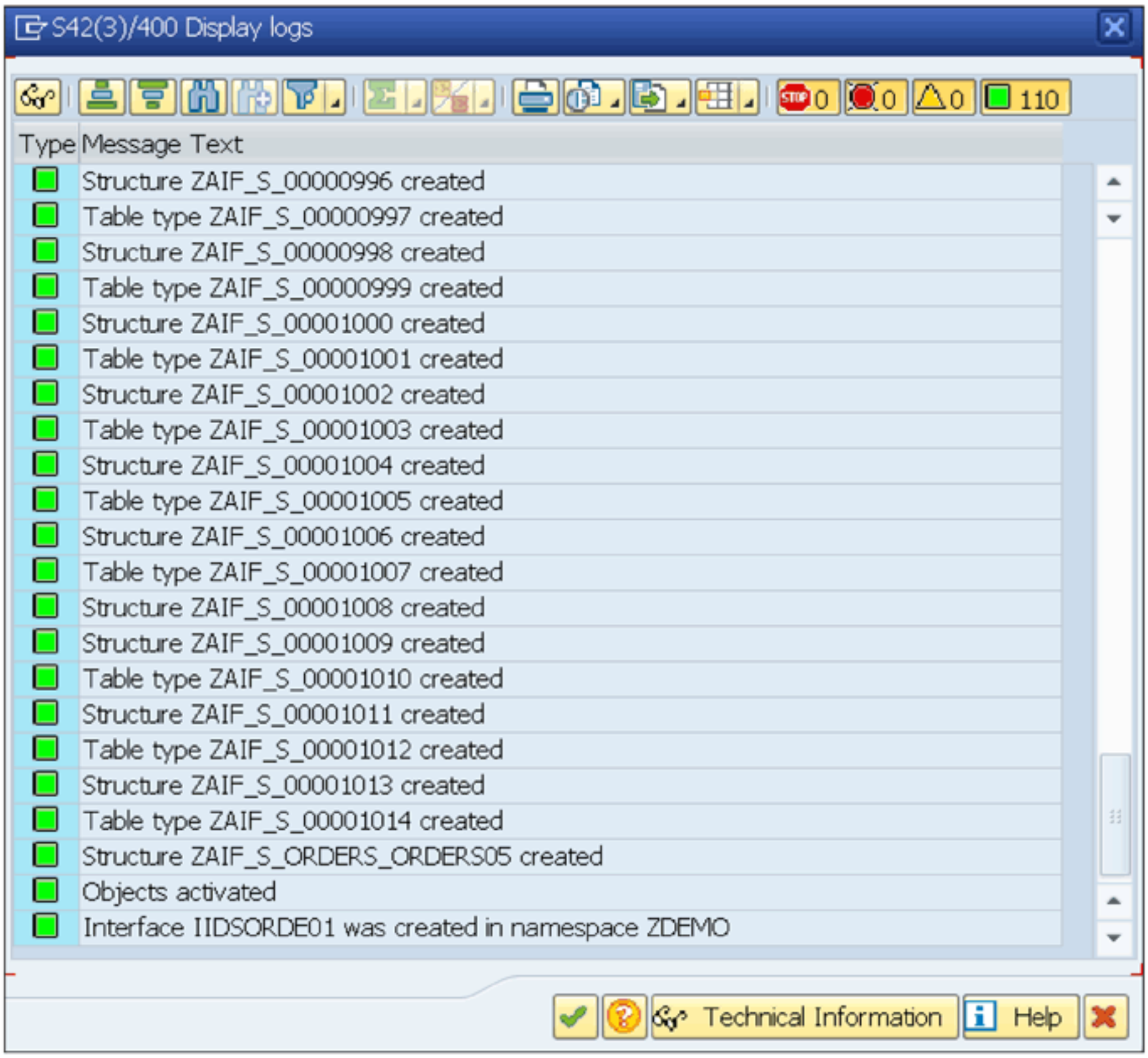

Figure 3 Log with Information about Generated ABAP Structure

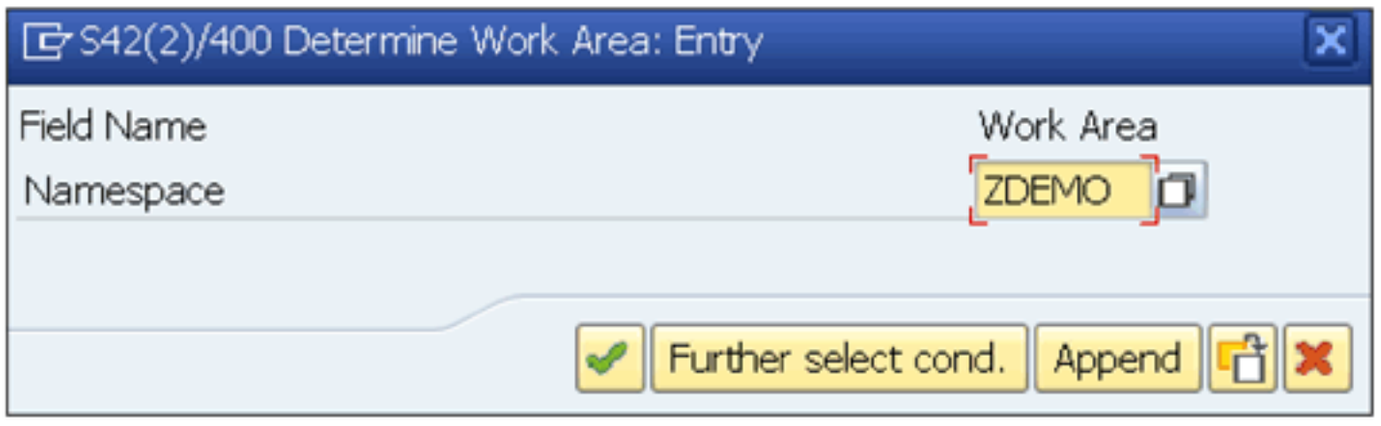

Figure 4 Namespace Determination

In the new window, double-click the **IIDSORDE01** interface to open another new window with all the information about the interface. In the **Raw Data**

Structure field, you can see the created structure. You can copy this structure to the **SAP Data Structure** field and then check the **Move Corresponding Structures** checkbox, as shown in Figure 5.

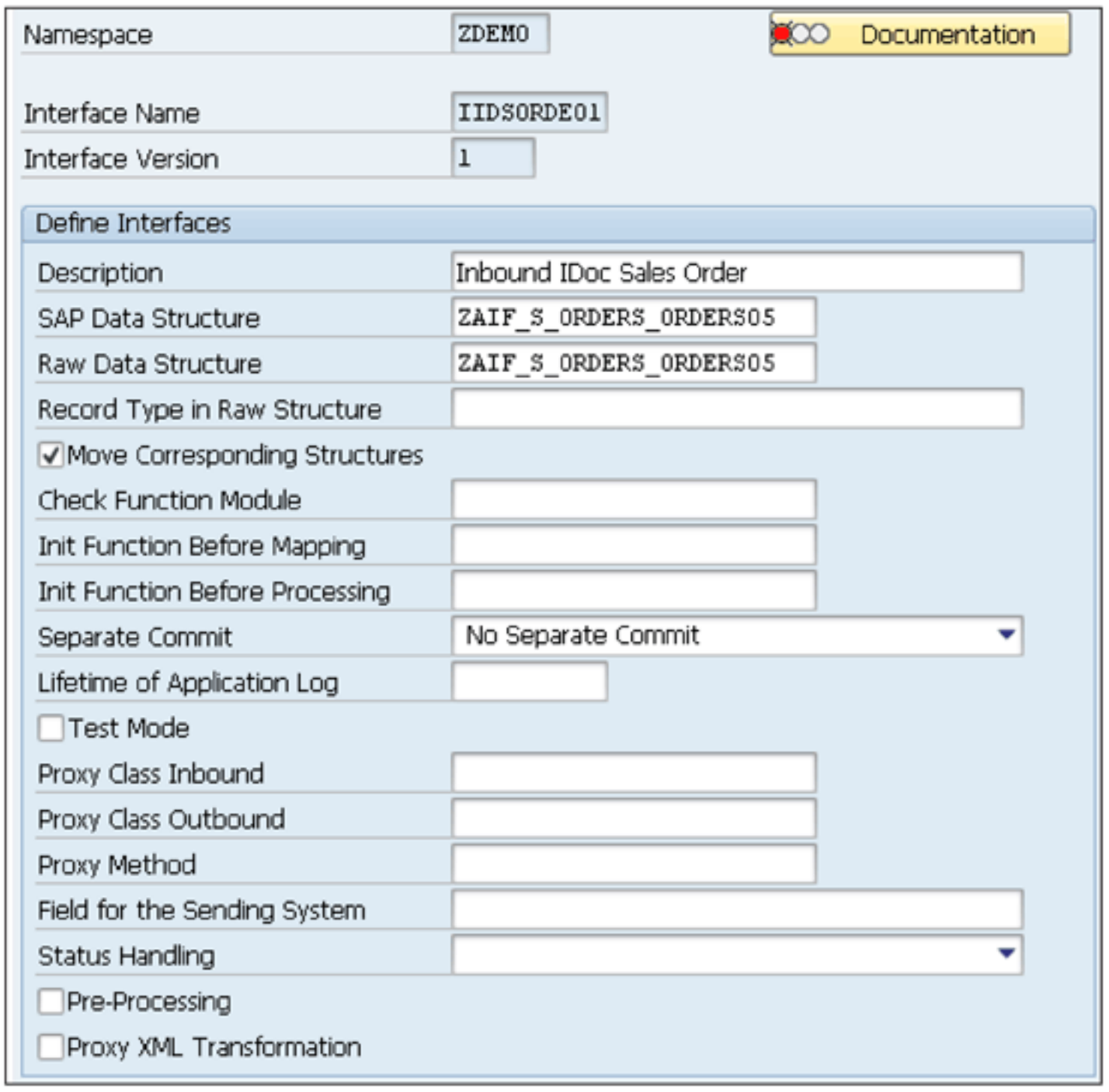

Figure 5 Details of ZDEMO/IDSORDE01/1 Interface

Then go back to the customization area and select the **Interface Development • Additional Interface Properties • Specify Interface Engines** node. Here you have to define engines that need to be assigned to the specific interface. For our interface, set **IDoc** for **Application Engine** and **Persistence Engine**, **AIF Index Tables** for **Selection Engine**, and **IDoc Status Records** for **Logging Engine**. The engine you use should be the same as that shown in Figure 6. Because we use SAP Application Interface Framework index tables as the selection engine, we will be able to create SAP Application Interface Framework alerts and we will be able to search based on key fields in monitoring and error handling.

Namespace	ZDEMO
Interface Name	IIDSORDE01
Interface Version	1
Define Interfaces (Engine Fields)	
Description	Inbound IDoc Sales Order
Application Engine	1 IDoc
Namespace	
Customer Engine	
Persistence Engine	1 IDoc
Namespace	
Customer Engine	
Selection Engine	0 AIF Index Tables
Namespace	
Customer Engine	
Logging Engine	1 IDoc Status Records
Namespace	
Customer Engine	

Figure 6 Engines Definition for IDoc Interface

In the last step, you have to confirm that proper IDocs are assigned to your interface. To do so, in Transaction /AIF/CUST go to **Interface Development • Additional Interface Properties • Assign IDoc Types** and make sure that you use **ORDERS** for the **Message Type** and **ORDERS05** for the **Basic Type**, as shown in Figure 7.

Change View "Define Interfaces (IDoc fields)": Details

New Entries

Namespace	ZDEMO
Interface Name	IIDSORDE01
Interface Version	1
Define Interfaces (IDoc fields)	
Description	ORDERS05
Message Type	ORDERS
Basic type	ORDERS05
Extension	
Message Variant	
Mess. function	

Figure 7 IDoc Message Type and Basic Type Assignment to Interface

2 Monitoring and Error Handling in Transaction /AIF/ERR

In the error handling Transaction /AIF/ERR you can monitor all SAP Application Interface Framework messages. In this place all messages for all different interfaces with their statuses appear. Each message belongs to a specific interface or a namespace. In error handling transaction you can see statuses, structures and content of the messages. In addition each message has its own log message. In Transaction /AIF/ERR you can see the details of what is wrong with the message, you can customize the view in order to make error handling an easier task. In this chapter you will learn more about this transaction.

2.1 Overview

To open the **Monitoring and Error Handling** window, go to Transaction /AIF/ERR. All the messages belong to a specific interface and each interface belongs to a specific namespace. In the first screen of the transaction, you can see the fields and buttons shown in Figure 8.

The screen is divided into the following sections:

- **Application Selection**
  For **Application**, choose the application ID, which is **AIF** by default.
- **Application-Specific Selection**
  The **Namespace**, **Interface Name**, and **Interface Version** fields allow you to define the interface that you want to monitor specifically. Each message is assigned to a concrete namespace, interface, and version. You can monitor the messages for a specific namespace and interface, but you can also monitor it for all namespaces or all interfaces within a specific message. The **Message Class** and **Message Number** fields are the representation of a specific log message. Based on these fields, you can look for all messages with a specific message class and number.

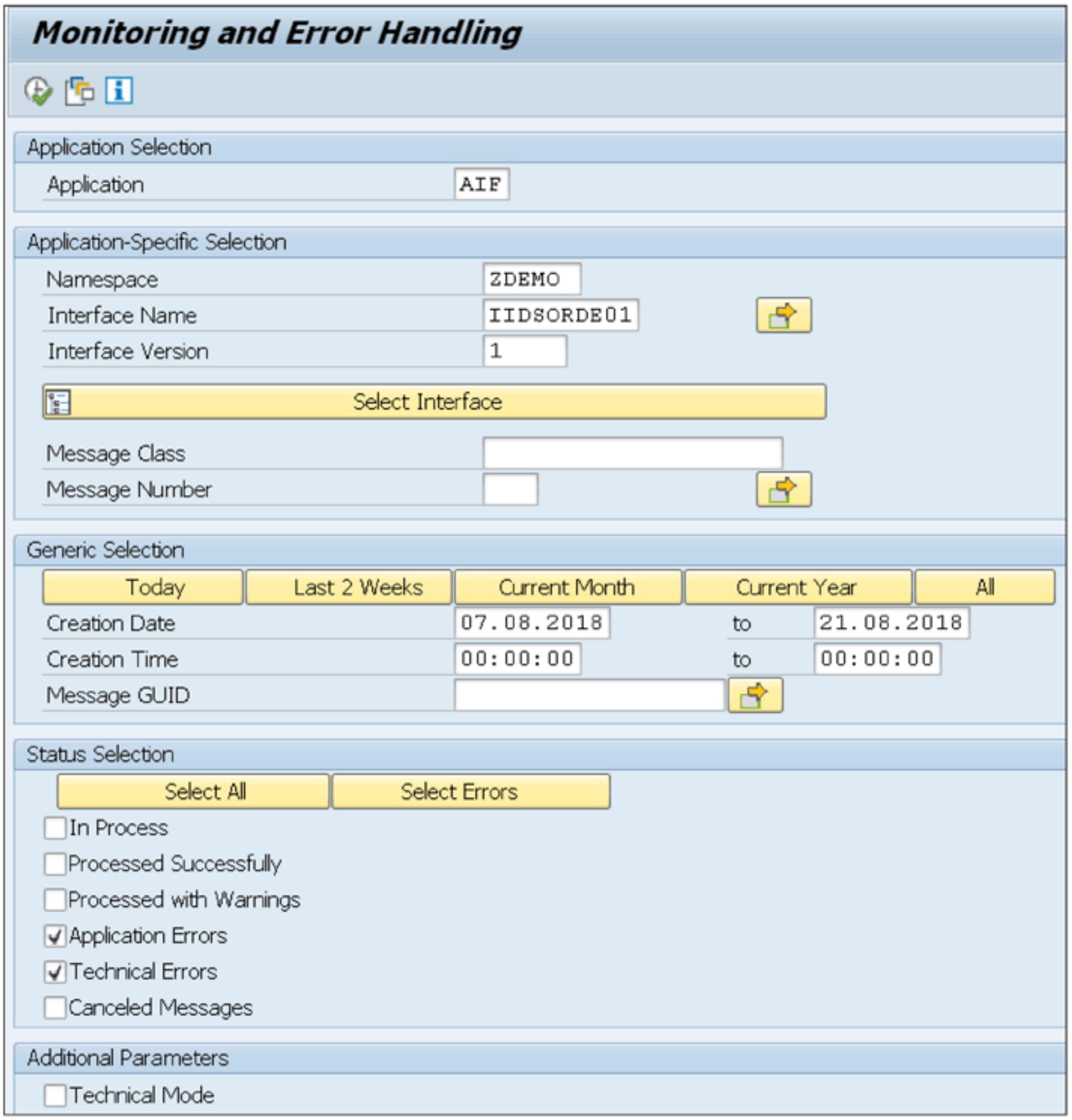

Figure 8 Monitoring and Error Handling Transaction

- **Generic Selection**

  In this section, you can provide the date and time range for which you want to monitor messages. In addition, you can provide the **Message GUID**, which is the unique message ID.

- **Status Selection**

  Here you can mark specific status-related checkboxes to display only messages with the specified statuses. You can choose messages that are

still processed, successful messages, ones with warnings, canceled ones, and messages with technical or application errors.

- **Additional Parameters**
  The **Technical Mode** checkbox allows you to run error handling in a specific mode, but this feature is not part of this E-Bite. The last field, **Max. Number**, specifies how many messages should be loaded when you press F8 or click the **Execute** button.

When you press F8 or click the **Execute** button, you will be redirected to next screen. This screen contains detailed information about every message. From here you can also cancel or restart messages. This screen is divided into four parts: at the top left is **Data Messages**, at the top right is **Log Messages**, at the bottom left is **Data Structure**, and at the bottom right is a **Data Content** view (see Figure 9).

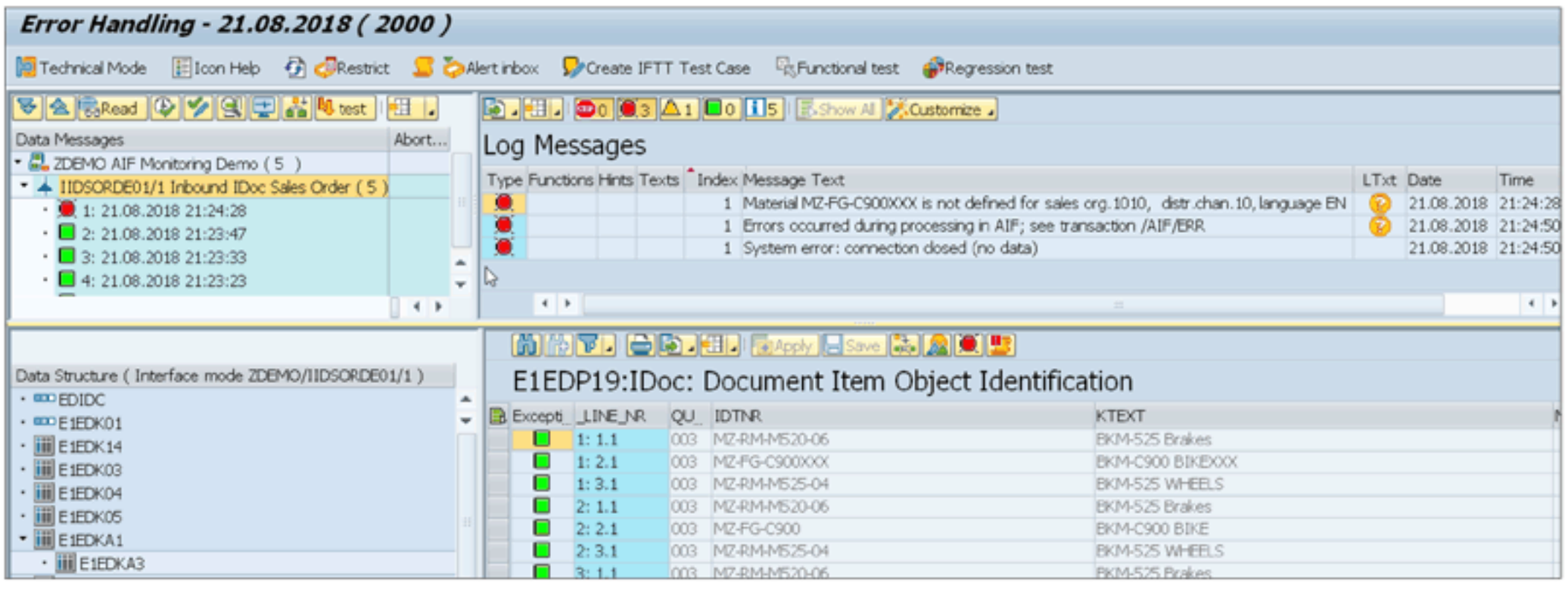

Figure 9 Error Handling Screen

Data Messages View

In the **Data Messages** area, you can see all messages with different statuses. All these messages are grouped by namespace, interface name, and version. From here you can perform additional actions for each message by using a specific button assigned in this section, as shown in Figure 10.

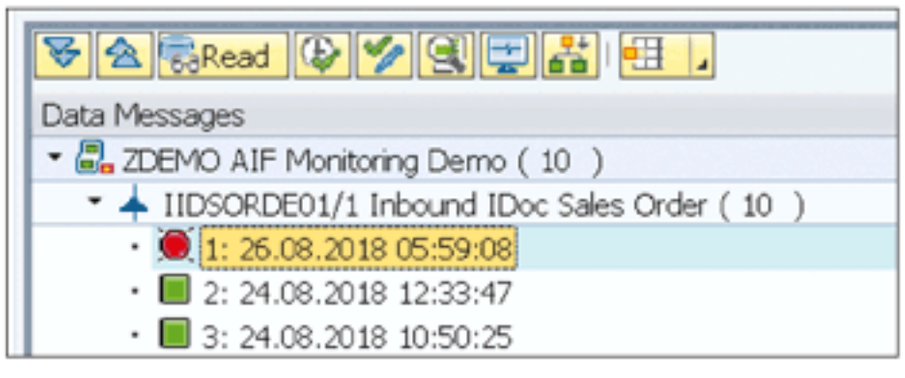

Figure 10 Data Messages View

The first two buttons allow you to expand and collapse all messages in a specific namespace and interface. Clicking the **Read** button collects information from the corresponding application log in the log messages and displays the structure for the selected message. The fourth button from the left, **Restart**, allows you to restart the selected message or the whole bunch of messages. You can restart messages in the foreground or in the background. The fifth button from the left, **Cancel**, cancels selected messages. Note that when you cancel a message, you won't be able to restart the message any longer. The next button, **Loop**, redirects you to the standard monitoring tool. In this E-Bite, we will work with IDoc messages, so this button will redirect you to the IDoc monitoring tool for the specific IDoc selected. The seventh icon from the left redirects you to the qRFC monitor. The last message in the group allows you to define a trace level for the selected message or the interface. If the message failed and you want more details about it, you can define a trace level and then restart the message. In the log messages, you can then see more detailed information.

Log Messages View

In this section, you can see all technical and application errors, information, warning, and success messages, with a specific icon representing each type of message, as shown in Figure 11.

Figure 11 Log Messages View

All these messages are saved when SAP Application Interface Framework processes the message. This section is helpful for error handling. In this view, you can create custom functions, hints, messages, and data links, as we will describe in detail later in Section 2.2.

Data Structure View

In this section, you can preview the whole raw structure of the selected message, as shown in Figure 12.

Figure 12 Data Structure View

When you double-click a specific structure or the table, the content of the selected node will be displayed in the data content view. In the next section, we provide a detailed explanation of how you can customize this view.

Data Content View

Once a specific node from the data structure is selected, the data content for this node is displayed in the data content view section. All the values are displayed in the external format, as shown in Figure 13.

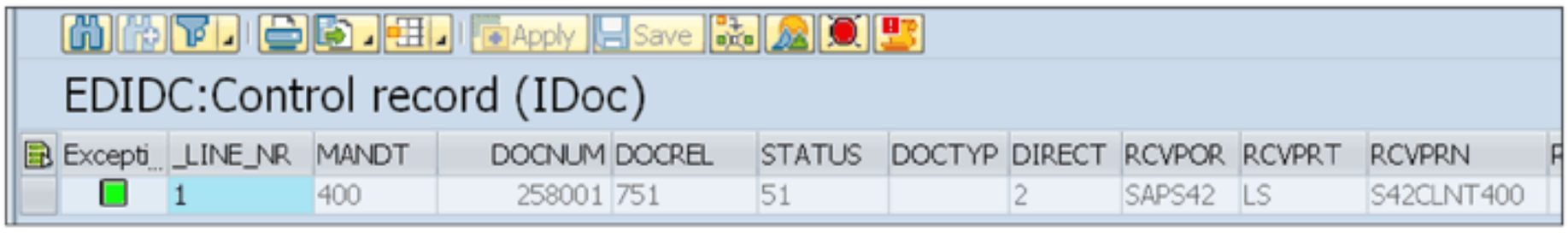

Figure 13 Data Content View

You can create a specific data link that connects the message from the log message to the value in the content view, as we will describe in Section 2.2.

2.2 Hints, Custom Texts, Functions, and Data Links

Custom functions, texts, hints, and data links in SAP Application Interface Framework enhance and speed up the process of monitoring messages and business documents related to the messages. They are easy to set up and, more important, business users almost immediately feel the difference of monitoring messages in SAP Application Interface Framework once they are ready to use. These features can be set for a concrete message ID and message number.

Concept and Setup

In this section you will learn what are the main features and benefits of each custom object. You will also learn how to set custom objects from Transaction /AIF/ERR.

Custom Functions

Custom functions allow you to define a specific action executed every time you select a custom function in the log messages view in the error handling transaction. With custom functions, you can call a specific transaction, report, or URL. Once you call your action, it's possible to transfer parameters from the messages that will fill the parameters of the transaction or the report. We will focus on creation of the custom function that will redirect us from the monitor to the sales order display related to this message in Transaction VA03. For this purpose, we will use standard message ID **V1** and message number **311** with message **Standard Order &1 Has Been Saved**. To create a custom function in the log message, first select the message for which you want to create a custom function, as shown in Figure 14.

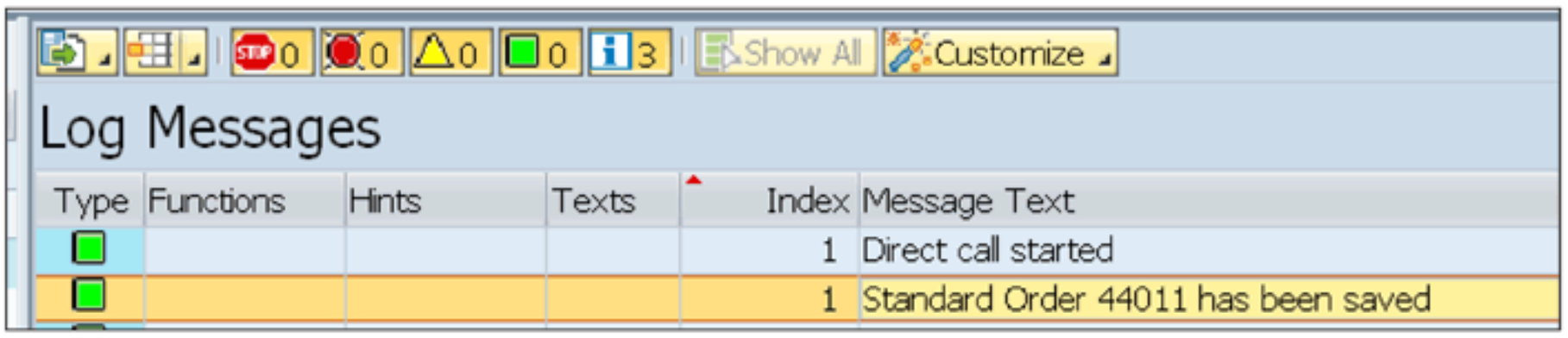

Figure 14 Messages in Log Messages

Next, from the menu above **Log Messages** choose **Customize • Custom Functions**, as shown in Figure 15.

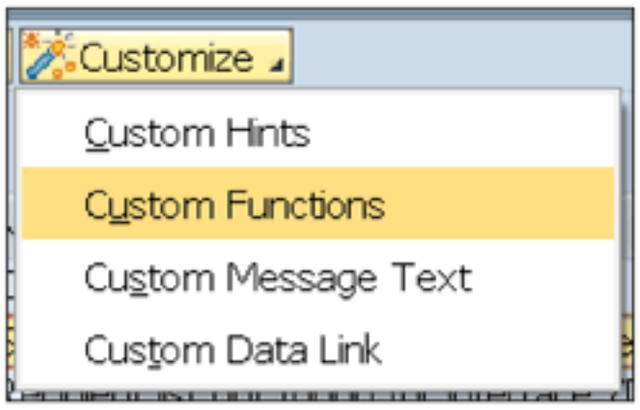

Figure 15 Customize Menu in Error Handling

In the opened window you will see three sections: **Message Details**, **Custom Function Action**, and **Custom Function Attributes**. In the **Message Details** section you can see the following entries, as shown in Figure 16, which are just for our information and cannot be changed:

- **Namespace**
  Namespace of your interface.
- **Interface**
  Interface name.
- **Interface Version**
  Version of the interface.
- **Message ID**
  ID of the message displayed in the log messages for which the custom function is being set.
- **Message No.**
  Number of message.

- **Message**
  Text of the message.
- **Logic System**
  Name of the logical system.
- **Message Var1–4**
  Variables used in the message.

Message Details

Namespace	ZDEMO	Logic System	
Interface	IIDSORDE01	Message Var1	Standard Order
Interface Version	1	Message Var2	44011
Message ID	V1	Message Var3	
Message No.	311	Message Var4	
Message	Standard Order 44011 has been saved		

Figure 16 Fields in Message Details Screen

In **Custom Function Action**, you can add the following type of actions:

- **Transaction**
  Code of the transaction used for redirection.
- **Report**
  Name of the ABAP program.
- **Screen Number**
  Screen number of the program, relevant only for reports.
- **URL**
  Address of the website.
- **Parameter** tab
  In this tab, all the parameters that have to be transferred to the transaction or program need to be defined:
  - **Parameter ID** is the parameter of the program or transaction.
  - **Parameter Description** is the description of the parameter, which is automatically filled once the transaction or report and parameter ID are filled.

- **Fill Method** defines how the parameter is filled in the transaction or program.
- **Manual Value** is the value that can be manually filled in the **Value** field of the **Parameter** tab. Once this method is chosen, the **Value** field becomes active.
- **Message Variable** is the value of the message variable defined for the message ID and number. Once this method is filled, the **Value from Message Variable** field is activated, allowing you to choose one of the message variables from the dropdown menu visible in the **Message Details** section.
- **Offset Message Variable** allows you to set the offset for the specific variable. When it is chosen, the related button is visible in the **Maintain** field. Once you click the button, a new window appears in which you can choose the message variable for which the offset will be set up in the **Message Variable** field, then in **Offset in Message Variable** you can set the offset, remembering that a value of 0 assigns the first character, value of 1 assigns the second character, and so on. In **Length of Offset**, you set how many characters you want to use as variables. You can see an example of setting up the offset in Figure 17.

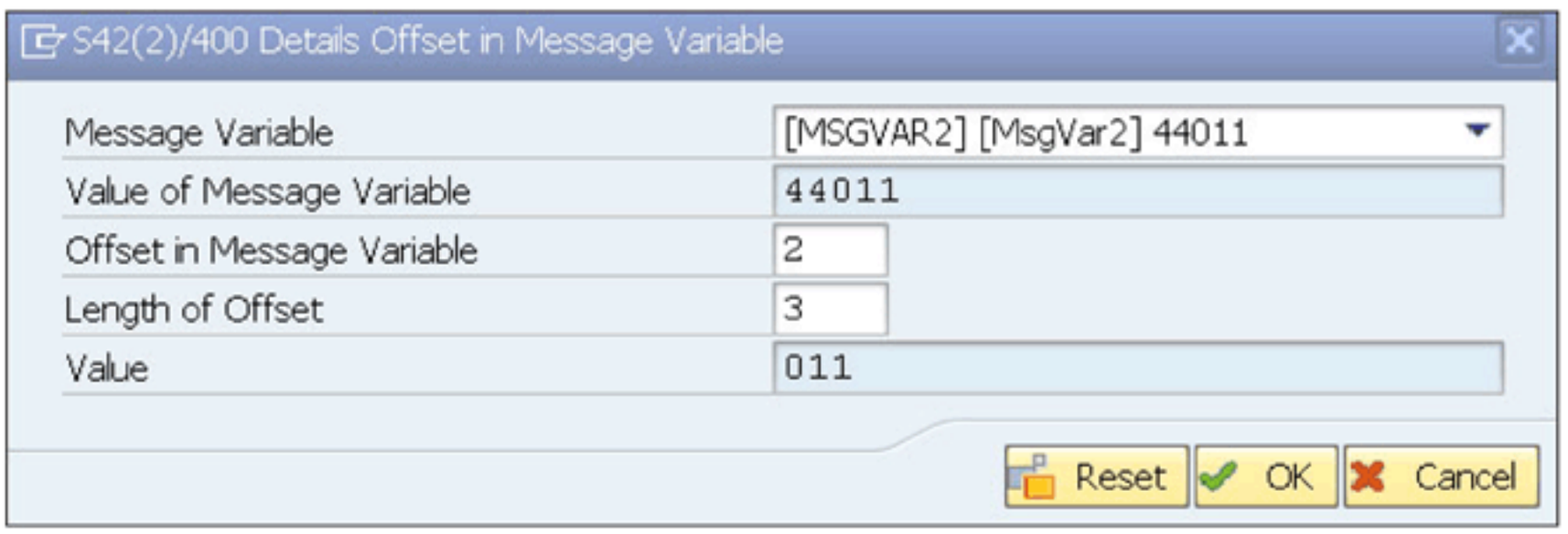

Figure 17 Details Offset in Message Variable

- **Value Mapping** lets you use variables in the message to perform previously created value mapping. Once this option is chosen, the **Maintain** button is visible; you can click it and assign existing value mapping by

providing the namespace and name of the value mapping and providing the parameters of the message.

In our example, in the **Transaction** field we type "VA03" to display the created sales order. In the parameter ID type "AUN", which is the sales order number parameter ID from Transaction VA03. As a **Fill Method** choose **Message Variable** from the dropdown menu, and in **Value from Message Variable** choose **MSGVAR2**, which is the order number. The result for the **Custom Function Action** section should be the same as shown in Figure 18.

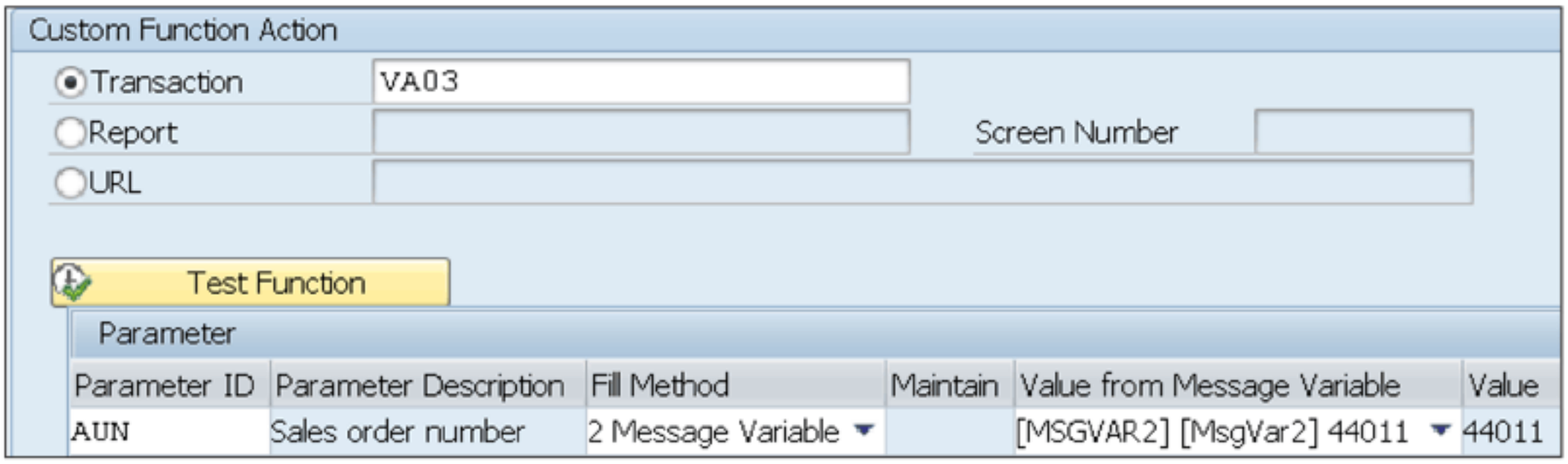

Figure 18 Custom Function Action

In **Custom Function Attributes**, the following information can be set:

- **Text**
  Used to provide information for what custom function can be used.
- **Tooltip**
  Contains information for what custom function can be used.
- **Language**
  Language of the custom function; the default language is English. You can use the **Maintain Other Languages** button to provide translations in other languages. First the message will be displayed in the logon language if it's maintained. If it's not maintained, then the second language set in the SAP system will be used. If this is not maintained, then the message will be written in English.

- **Icon**
  Code of the icon that will represent this custom function.
- **Set First Screen**
  Should be checked if you want to skip the first screen of the transaction or report.
- **Start New Session**
  Should be checked if a transaction or report should be called in a separate session.

In this case, type "Display Sales Order" for **Text** and **Tooltip** and type "@9F@" for **Icon**. Also select the **Skip First Screen** and **Start News Session** checkboxes. When your entries match Figure 19, click **Save**.

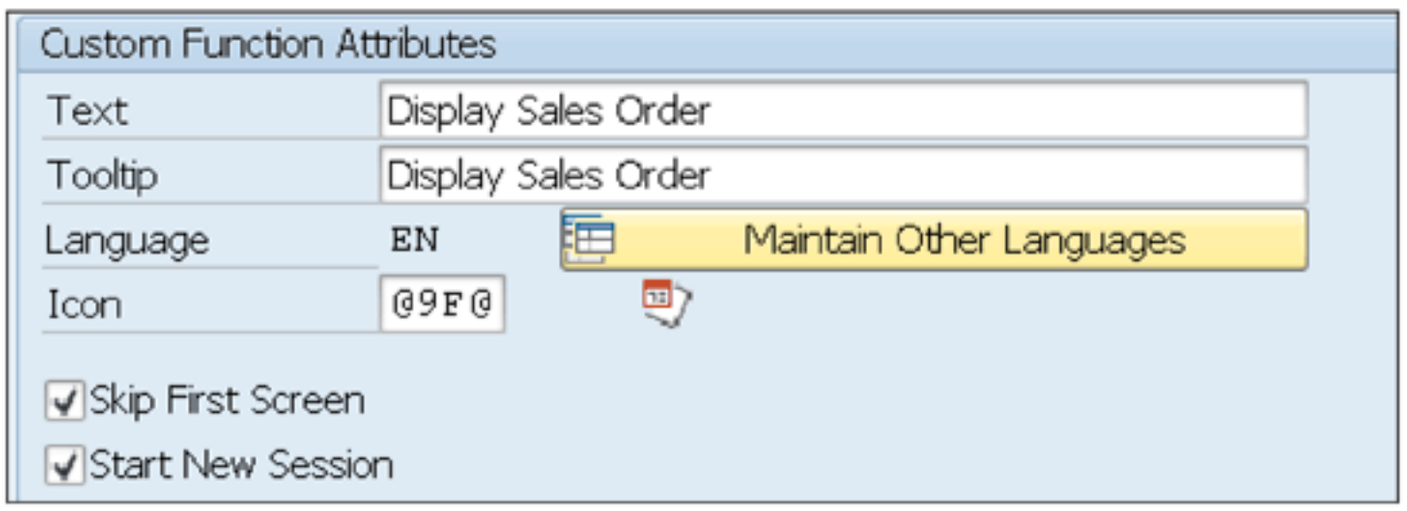

Figure 19 Custom Function Attributes

Now you should see the icon next to the message for which you set the custom function, as shown on Figure 20. Once you click on it, you will be redirected directly to Transaction VA03 and to the sales order, with the number visible in the message!

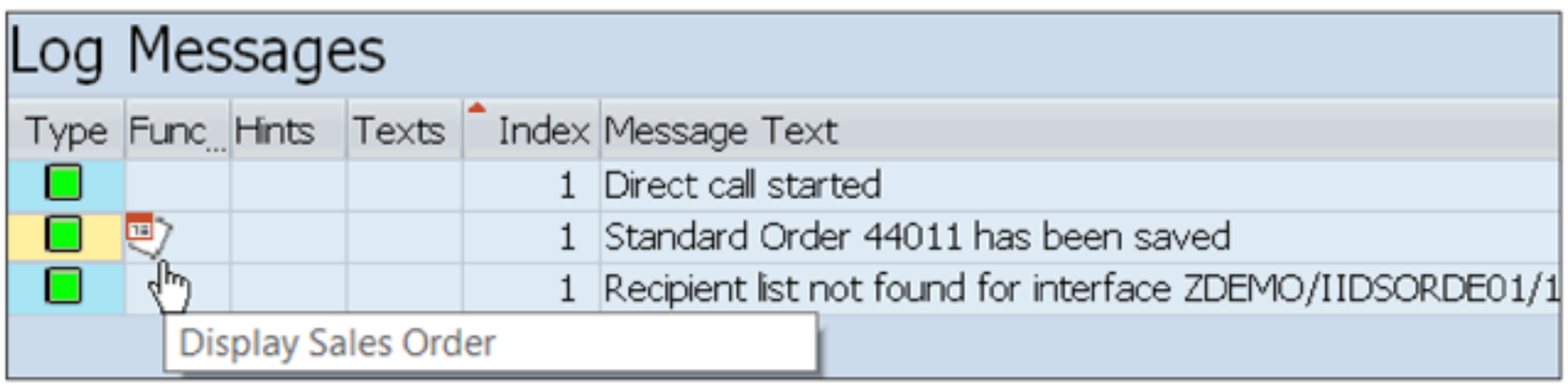

Figure 20 Icon Display Sales Order Assigned to Message Text

Custom Hints

Custom hints are very useful when you want to document in a few sentences the procedure based on the message ID and message number. They can be used for all types of messages, but they are the most useful for the messages with errors. Then you can define a hint to describe how the problem can be solved. In this scenario, we will send an incorrect message with the sales organization that is not maintained (customization for this sales organization is missing) in the system. The message will be sent through Transaction WE19 with value 1111 in the E1EDK14 segment with qualifier 008 in the organization field, as shown in Figure 21.

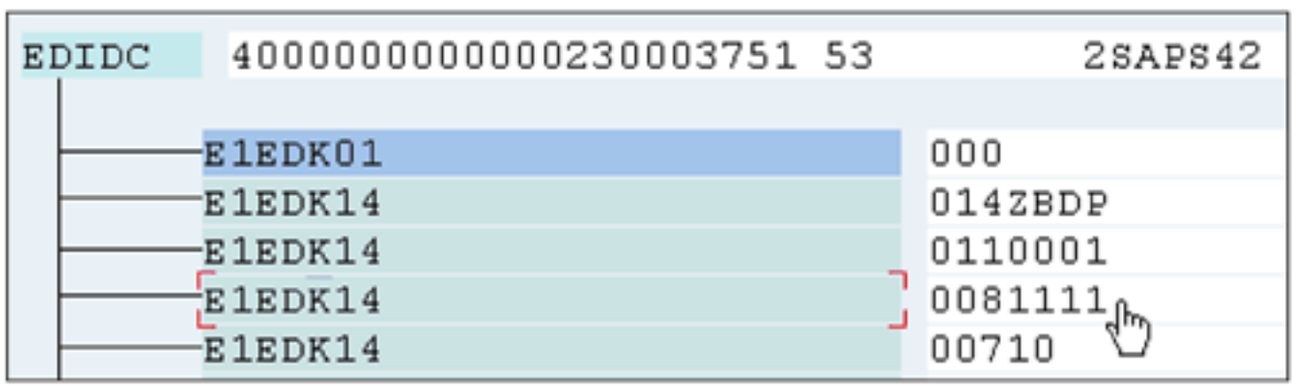

Figure 21 Transaction WE19 with Segment E1EDK14 Edited

Now in the SAP Application Interface Framework error handling, you should see a new message with error, as shown in Figure 22.


Log Messages

Type	Functions	Hints	Texts	Index	Message Text
				1	Sold-to party 10100002 not maintained for sales area 1111 10 00

Figure 22 Error Message in Log Messages

To create a new custom hint, select the message for which you want to create the hint and choose **Customize • Custom Hints** from the top menu. A new window will appear with three sections: **Message Details**, **Hint Attributes**, and **Enter a Text** will be displayed. The **Hint Attributes** section contains just a **Tooltip** field in which you can type a tooltip that describes the hint and a field to define the **Language** of the tooltip. In the **Enter a Text** box, you can write or upload from a text file hints displayed for this specific error. In this

case, we want to create a custom hint for a message ID VP and message number 197. The standard message says that **Sold-To Party Is Probably Not Maintained for the Sales Area 1111 10 10**, but it doesn't contain any information indicating that the sales organization is not maintained. Provide a procedure in your hint, as shown in Figure 23, and save the entry.

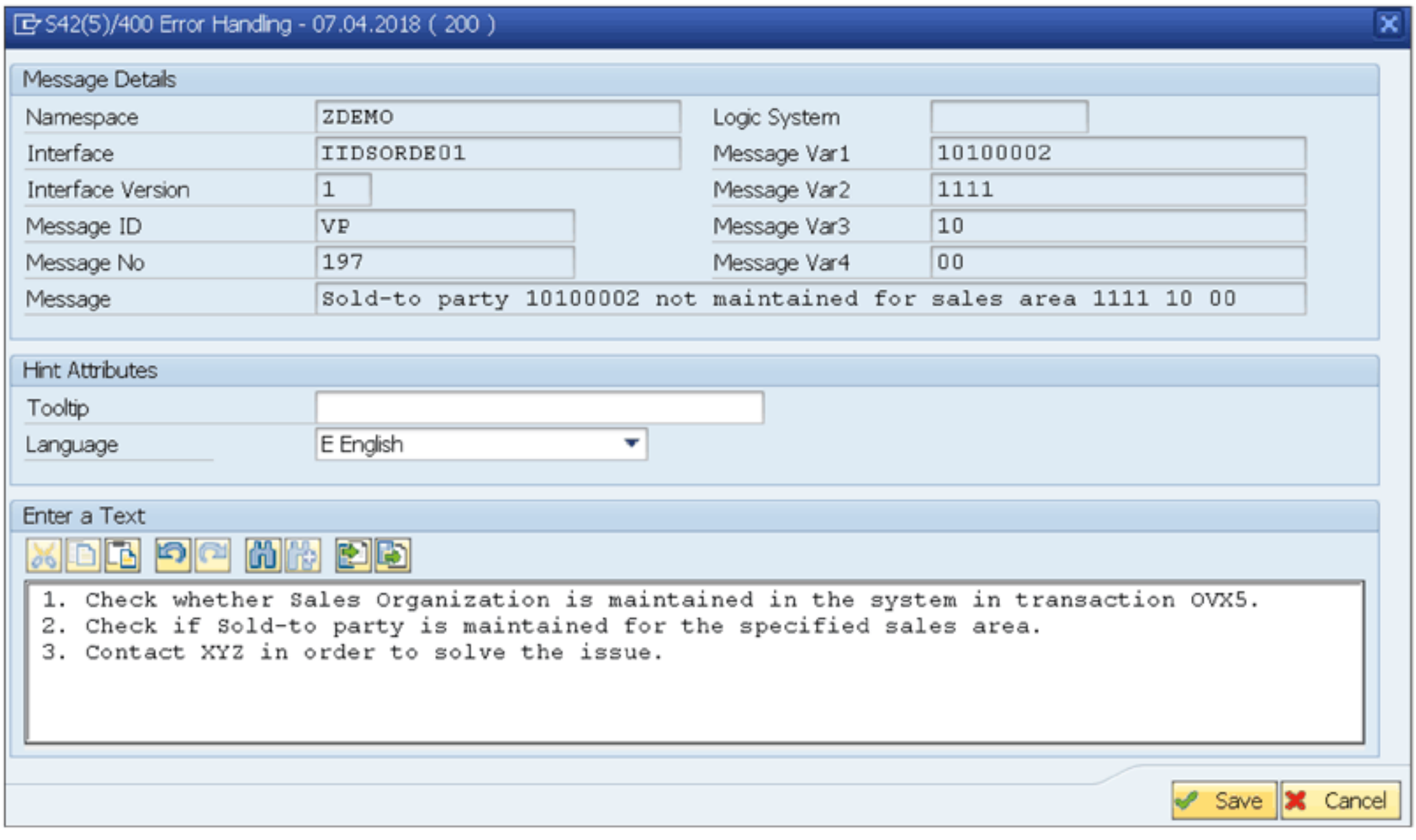

Figure 23 Screen Showing Custom Hints Creation

Now, next to the message in the **Hints** column, a new icon should be visible, as shown in Figure 24. Once you click it, a new pop-up with the procedure will be displayed. You can define as many hints as necessary.

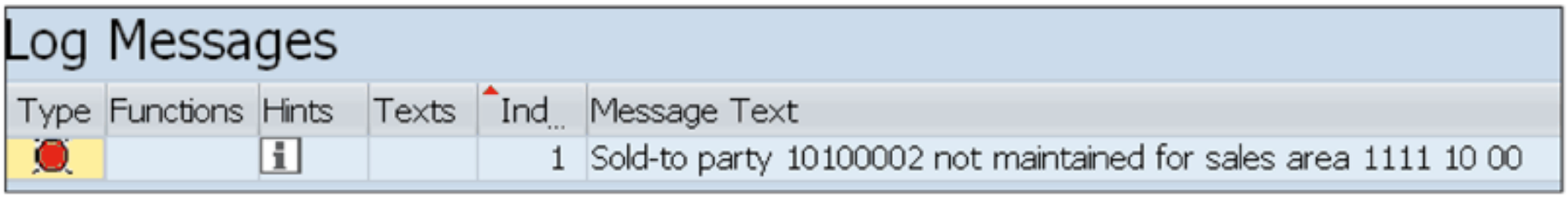

Figure 24 Log Messages Extended with Custom Hint

Custom Texts

Custom texts in SAP are used to change the standard text for a specific message ID and message number. This is very helpful if you have some tailor-made solutions or you just want to see a concrete text immediately in the **Message Text** column in **Log Messages**. For the purposes of this E-Bite, we will create custom text for the same message for which we created a custom hint. To create custom text, select the message for which you want to create the text, then choose **Customize • Custom Hints** from the menu. In the new window, you can see two sections: **Message Detail** and **Custom Message Text**. In the **Custom Message Text** section, you can provide **New Message** text and the **Language**. Note that you can type only text and variables here. To display variables visible in the message detail section, use *&1*, *&2*, *&3*, and *&4* symbols. In our custom message text, we want to indicate that the sales organization or sold-to party is not maintained. For this purpose, we will extend the text with variable *&2*, representing the sales organization, as shown in Figure 25.

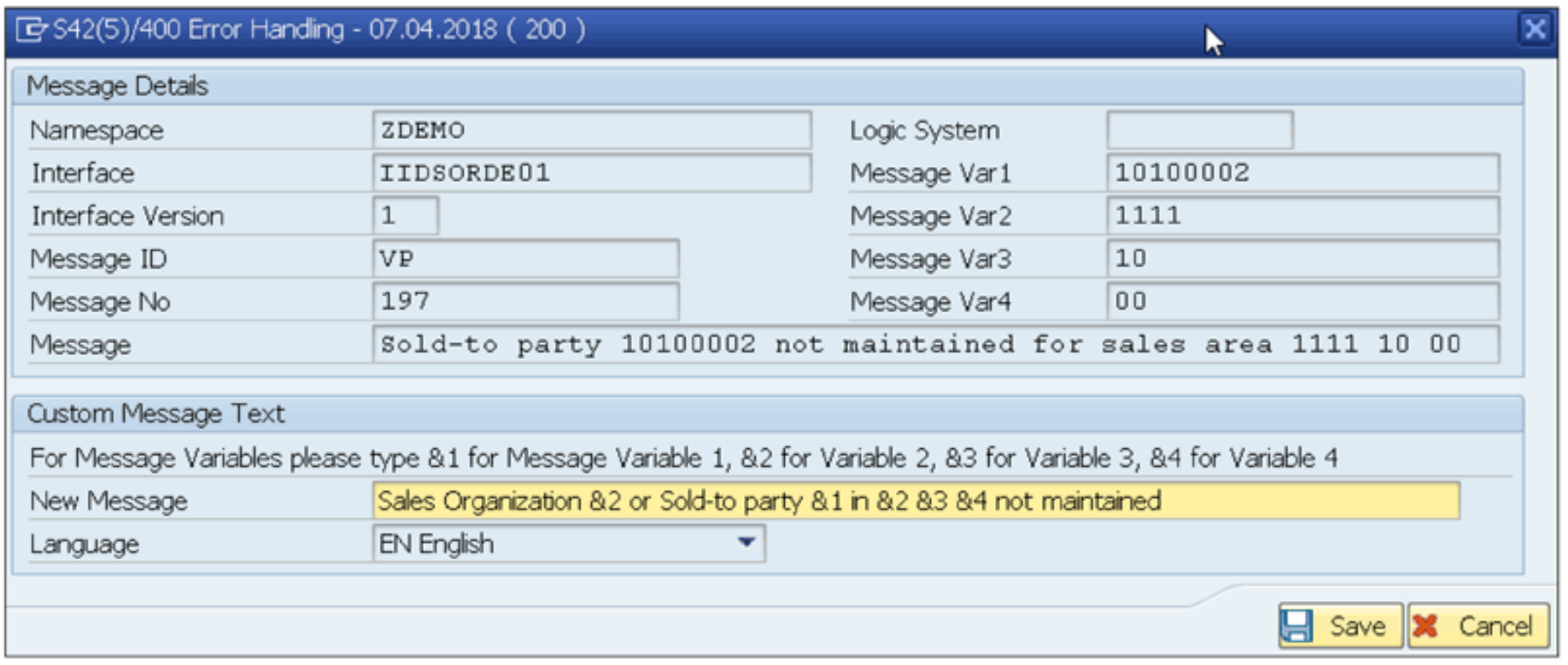

Figure 25 Screen Showing Creation of Custom Text

Save the entry; the new message text should be displayed in the log messages, as shown in Figure 26. In addition, in the **Texts** column you should see a new icon; thanks to this, you will know immediately whether the text is standard or custom, because only custom texts have the additional icon.

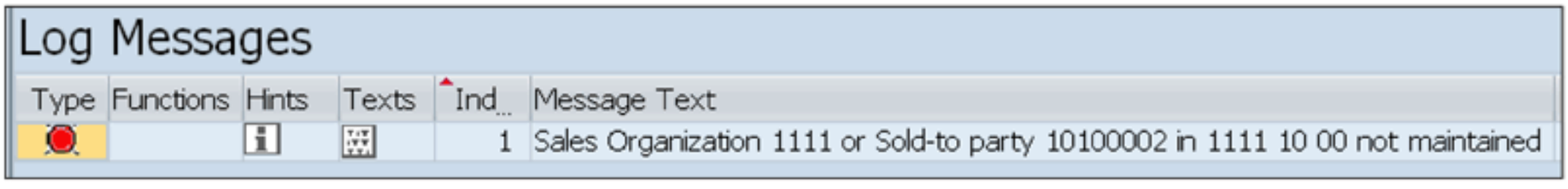

Figure 26 Log Messages View with Custom Text

Custom Data Links

Custom data links allow you to create a link between a message in the log messages and a specific field from the data content. When a data link is created properly, once you click the specific message, the field that was set is shown in the data content. In addition, when the message is in an error status, the field that is linked has a red background. Data links are helpful for both successful and error messages because they save time and prevent you from needing to look for a field in the whole structure.

For the purposes of this E-Bite, we will create a data link between a standard SAP error message, **No Customer Master Record Exists for Sold-To Party**, and the PARTN field from the E1EDKA1 structure. To link such a message, we will send a new message from Transaction WE19 with value 10100008 in the partner field of the E1EDKA1 structure, as shown in Figure 27, because we know that in our system partner 10100008 is not maintained.

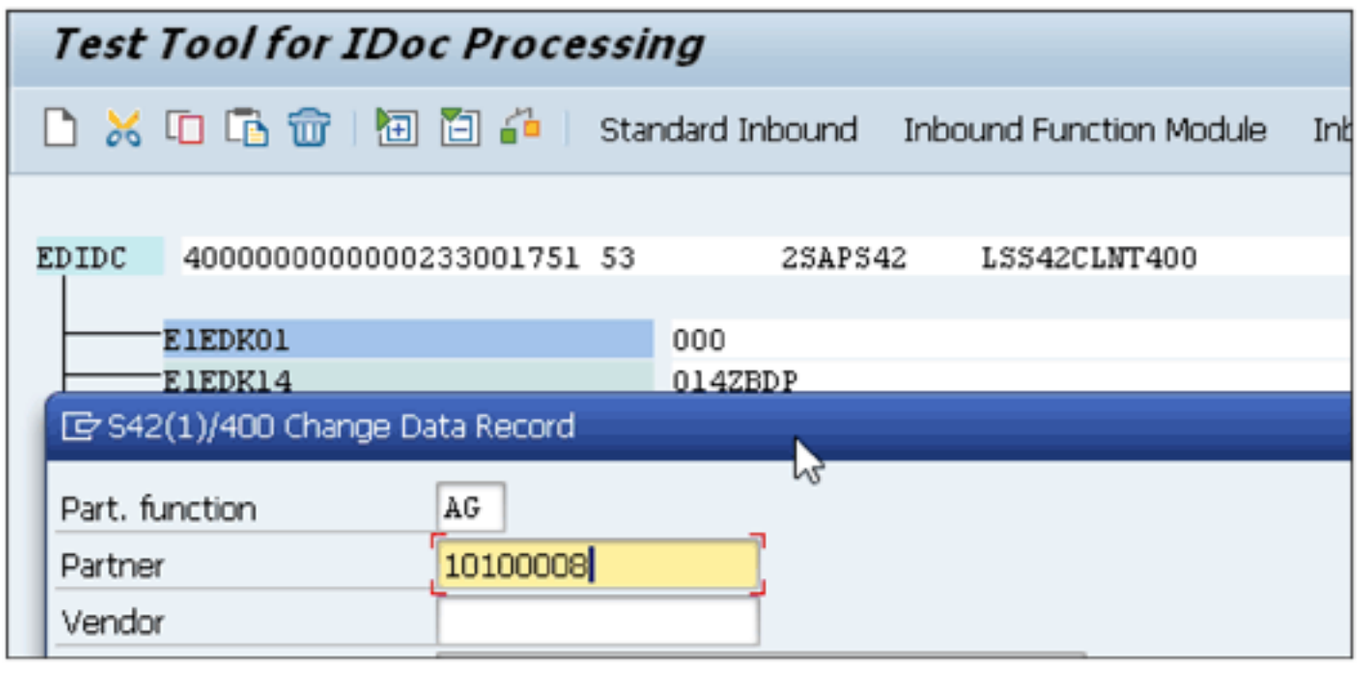

Figure 27 Transaction WE19 with Filled Partner

Now, in SAP Application Interface Framework error handling, we should have a new message in an error status. To create a data link, open node

E1EDKA1 in the data structure. Then select only content of the message in the log messages, and next select only the content of the PARTN field of the E1EDKA1 structure, as shown previously in Figure 22.

Tip

Creation of data links requires a bit of precision. Make sure that you only select the column **Message Text** in the log messages instead of the whole line of the message, including functions, hints, and so on. The same is true for fields in the data structure: do not select the whole line or column, but select just the field with content (in this case, 10100008), exactly as shown in Figure 28.

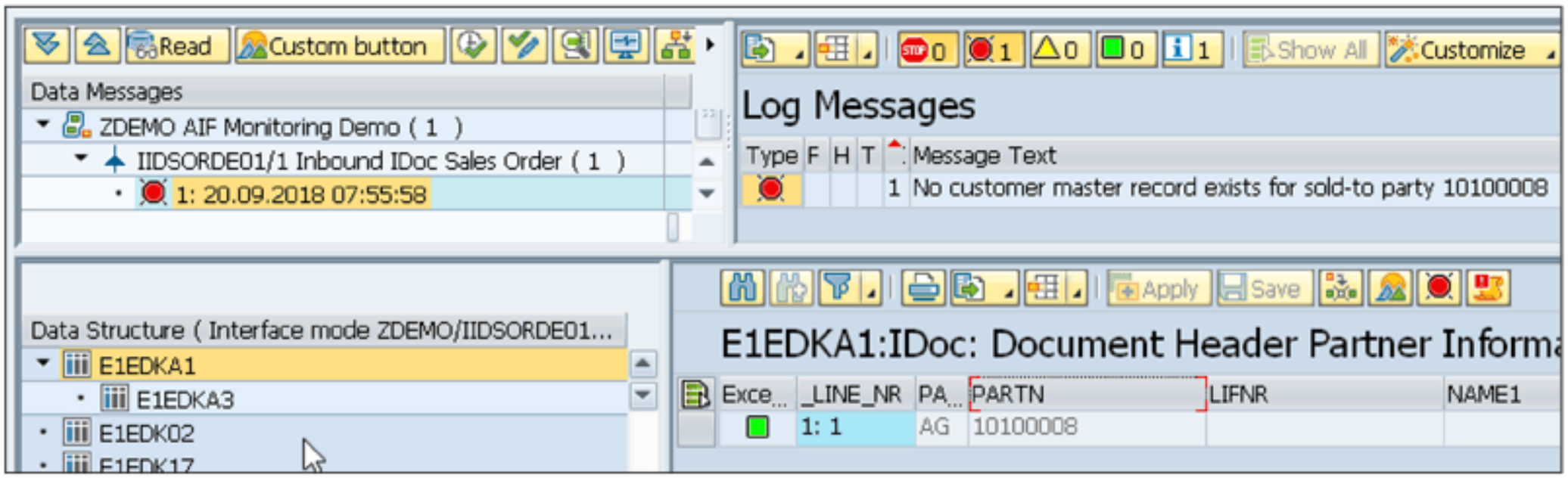

Figure 28 Data Link between Message and Partner Number

Once you have selected the content of the message text and content of the PARTN field, choose **Customize • Custom Data Links** from the menu. A new window will appear as shown in Figure 29. Click the **Create** button.

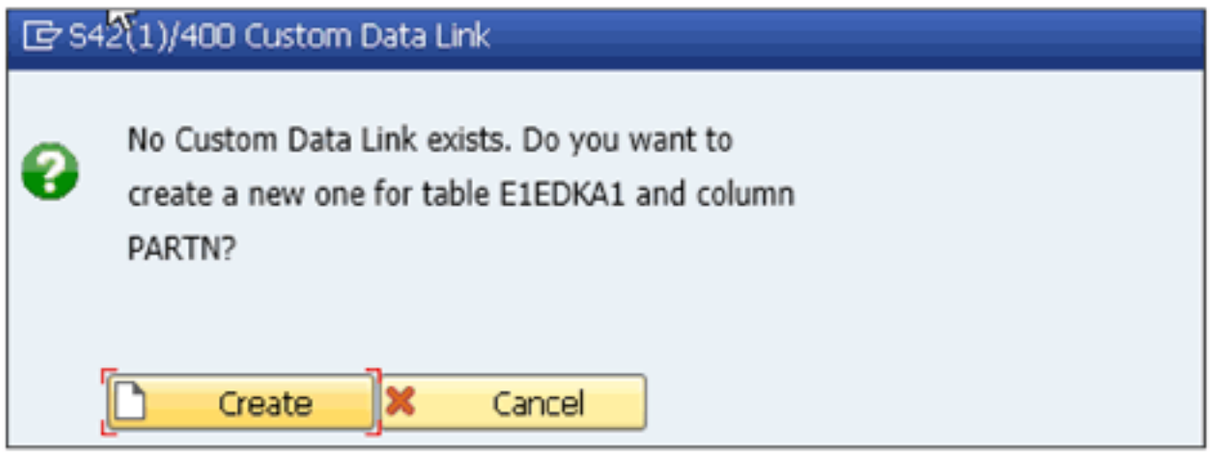

Figure 29 Creation of Data Link

Now you can double-click the message in the log messages; immediately, the PARTN field will be shown, marked in red because the message is in an error status (see Figure 30).

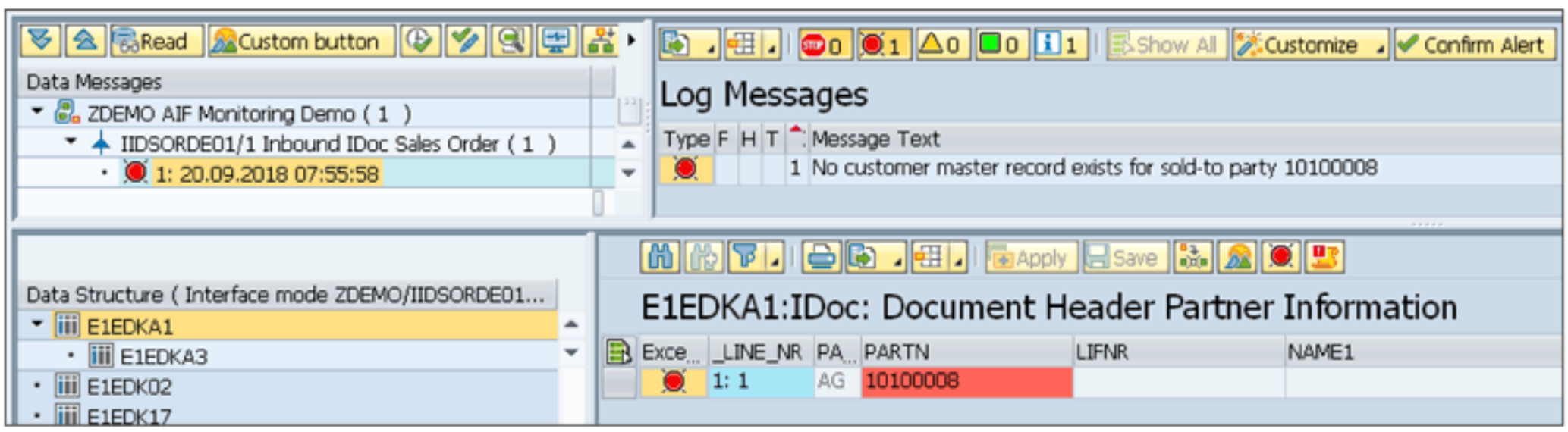

Figure 30 Completed Data Link

Administration

In the previous subsections, you learned how to create custom texts, functions, hints, and data links. It is important to know that once you create custom objects directly from the SAP Application Interface Framework error handling screen, only you (or the person who created it) can see them. To create custom objects visible for all users or a list of selected users, it is necessary to maintain them in administration mode using specific transactions. Each custom object has to be maintained in a different transaction. There are two approaches for creating custom objects. The first approach is to create custom objects directly from Transaction /AIF/ERR and then maintain them in the administration mode; the second approach is to create them from scratch in the administration mode. In this E-Bite, we will follow the first approach, which we think is a more convenient option.

Custom Functions Administration

To maintain custom functions in administration mode, open Transaction /AIF/CUST_FUNC. A new window should appear in which you have to provide details of the interface, as shown in Figure 31, and then press Enter or click the green checkmark button.

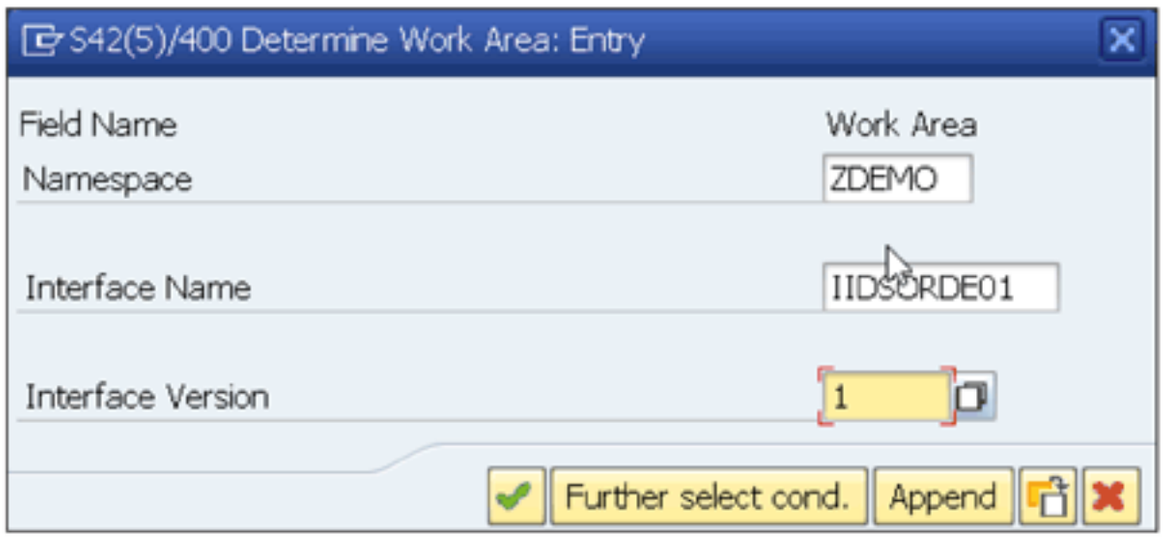

Figure 31 Interface Determination

You will see a list of all custom functions created by all users, like that shown in Figure 32.

Namespace	ZDEMO
Interface Name	IIDSORDE01
Interface Version	1

Define Custom Functions

Function ID	Function Text	Function Tooltip	Icon	Log.System	Message class	Msg.	User
12B17D7689C01ED88EC81070A818A20F	Display Sales Order	Display Sales Order	@9F@		V1	311	MNOWAK
12B17D7689C01ED8938B821045E1A219	VA03	VA03			V1	311	KLUKA

Figure 32 List of Existing Custom Functions

Open the function you created in Transaction /AIF/ERR by double-clicking it. You will see a new window (Figure 33).

The following fields are filled with the information you provided in Transaction /AIF/ERR while creating your custom function: **Funct. Txt.**, **Fct. Tooltip**, **Function Icon**, **Logical System**, **Message ID**, **Message Number**, **Skip First Screen**, **Start in New Session**, **Cust. Func. Type**, and **URL**.

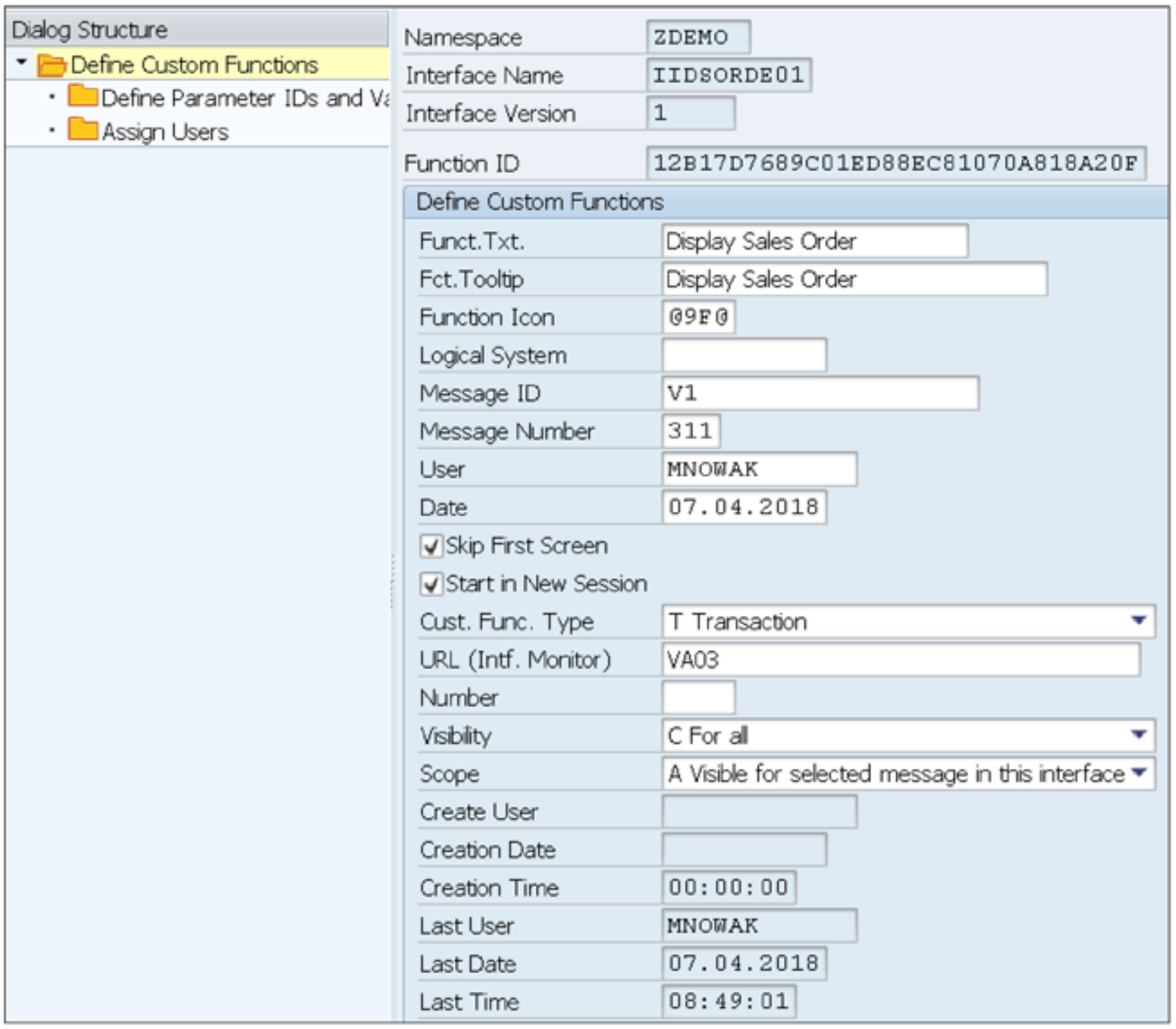

Figure 33 Custom Function Maintenance

There is some additional information provided, like the **User** who created the custom function and the **Date** the custom function was created. There are also two important pieces of information you have to provide:

- **Visibility**
  In this dropdown field, you have to choose who should see this custom object. Three options are available:
  - **For All**
    All users will see this custom object.
  - **Just for Current User**
    Only user determined in **User** field will see this custom object.

– **For List of Users**
  Only users who are determined in **Assign Users** node will see this custom object (see Figure 34).

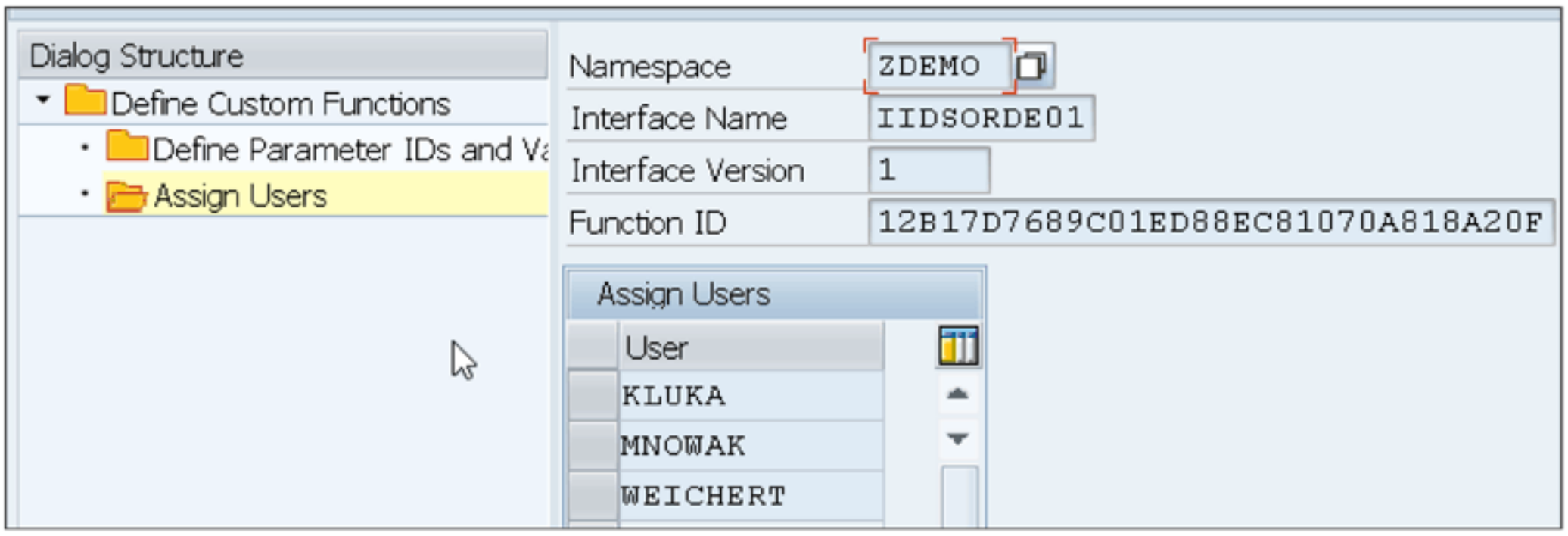

Figure 34 User Assignment for Custom Function

- **Scope**
  In this dropdown field, you have to choose for which messages and interfaces the custom object will be visible. The following options are available:
  – **Visible for Selected Messages in This Interface**
  – **Visible for All Messages in This Interface**
  – **Visible for Selected Messages in All Interfaces**
  – **Visible for All Messages in All Interfaces**

For the purposes of this E-Bite, set the **Visibility** to **For All** and the **Scope** to **Visible for Selected Messages in All Interfaces**. This way, a custom function we created for the specific message will be reused in other interfaces. In the **Define Parameter IDs and Values** node, you can see and maintain parameters and IDs you set in Transaction /AIF/ERR, as shown in Figure 35.

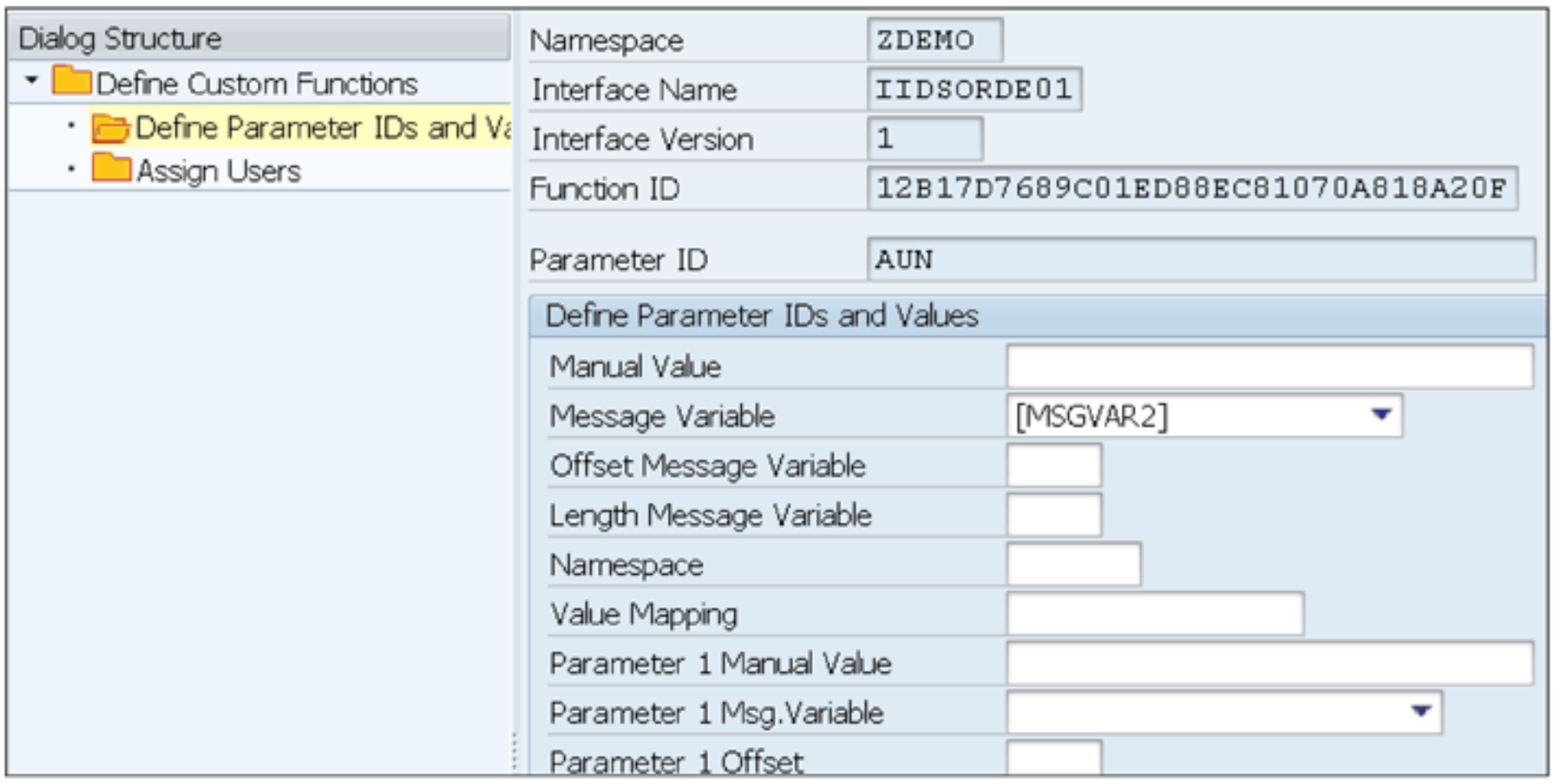

Figure 35 Custom Function Definition in Transaction /AIF/CUST_FUNCTION

Custom Hints Administration

To manage custom hints in administration mode, go to Transaction /AIF/CUST_HINTS. In the new window, provide details of your interface as shown in Figure 31, then press Enter. In the new window, there is a list of existing custom hints (see Figure 36).

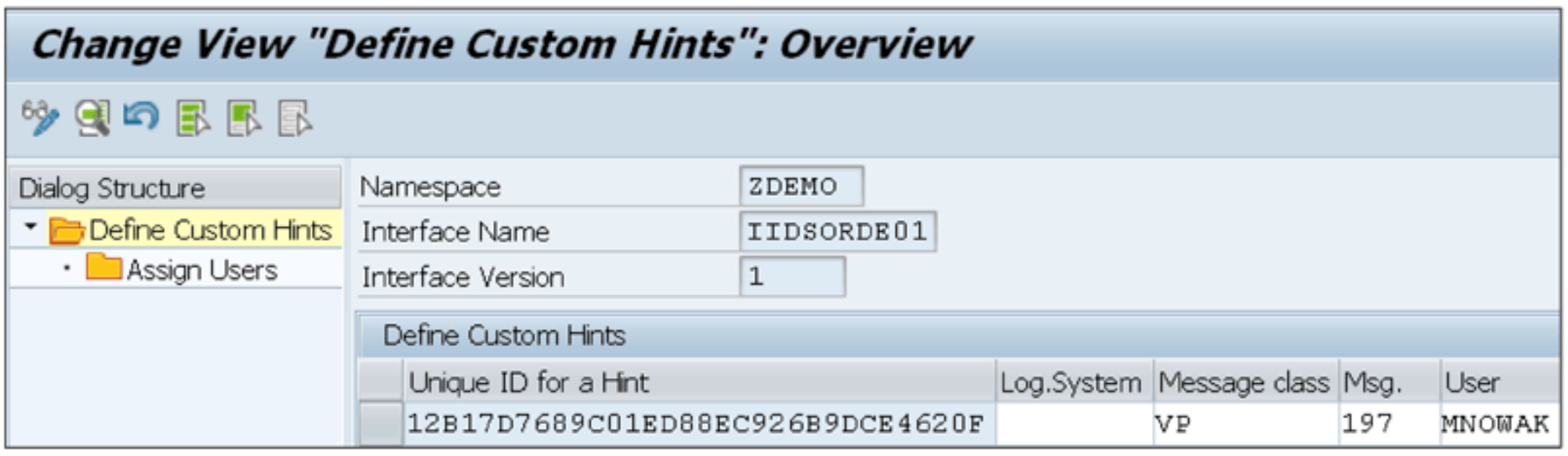

Figure 36 Custom Hints Definition in Transaction /AIF/CUST_HINTS

Open the hint for message class VP and message number 197, which you created in Transaction /AIF/ERR. As shown in Figure 37, the following fields are filled with the information you provided during custom hint creation: **Message ID** and **Message Number**. Other fields like **User**, **Date**, and **Number** are

filled automatically. The **Number** field is just a representation of the screen number used in Transaction /AIF/ERR.

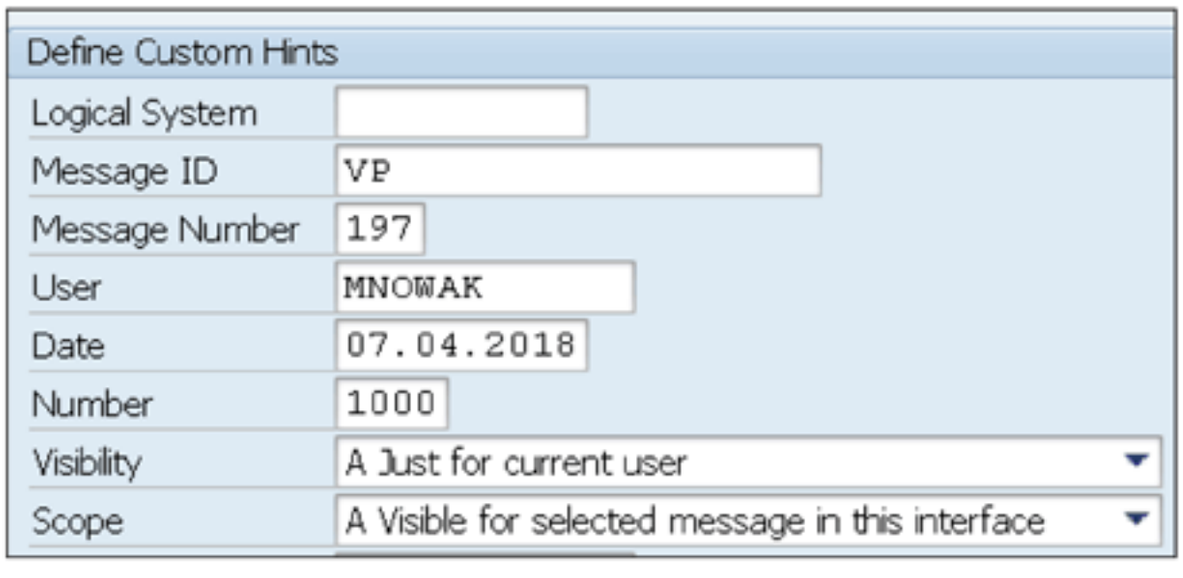

Figure 37 Custom Hints Definition

Two additional fields, Scope and Visibility, have exactly the same purpose and behavior as we described for custom functions. Set **Visibility** to **For All** and **Scope** to **For Selected Message in All Interfaces**. You can also assign users that should see this specific hint, as was shown in Figure 34.

Custom Texts Administration

To administer custom texts, open Transaction /AIF/CUST_TEXT. In the new window, provide details of your interface, as was shown in Figure 31. You will see a new screen with a list of the existing custom texts, as shown in Figure 38.

Figure 38 List of Existing Custom Texts

Open the custom text you created for message class: VP and message number 197 by double-clicking it. In the new window, you will see all the information you provided during custom text creation in Transaction /AIF/ERR (Figure 39). In addition, you will also see **Scope** and **Visibility**, as described in

previous subsections. Set **Visibility** to **For All** and **Scope** to **For Selected Message in All Interfaces**.

Namespace	ZDEMO
Interface Name	IIDSORDE01
Interface Version	1
Custom Text ID	12B17D7689C01ED88EC9911E9452220F
Define Custom Message Text	
Logical System	
Message ID	VP
Message Number	197
Visibility	A Just for current user
Scope	A Visible for selected message in this interface
Message Text	Sales Organization &2 or Sold-to party &1 in &2 &3 &4 not maintained
Create User	MNOWAK
Creation Date	07.04.2018
Creation Time	10:15:04
Last User	MNOWAK
Last Date	07.04.2018
Last Time	10:17:06

Figure 39 Custom Text Maintenance

If you want to set visibility only for selected users, you can add a list of users in the **Assign Users for Custom Message Text** node, as was shown in Figure 34.

Custom Data Links Administration

When you want to customize data links you created, open Transaction /AIF/CUST/LINK. In the new window, provide details of your interface, as was shown in Figure 31. You will see a new screen with a list of the existing custom data links. Open the custom data link you created for message class VP and message number 197 by double-clicking it. A new window will appear, as shown in Figure 40. Set **Visibility** to **For All** and **Scope** to **For Selected Message in All Interfaces**.

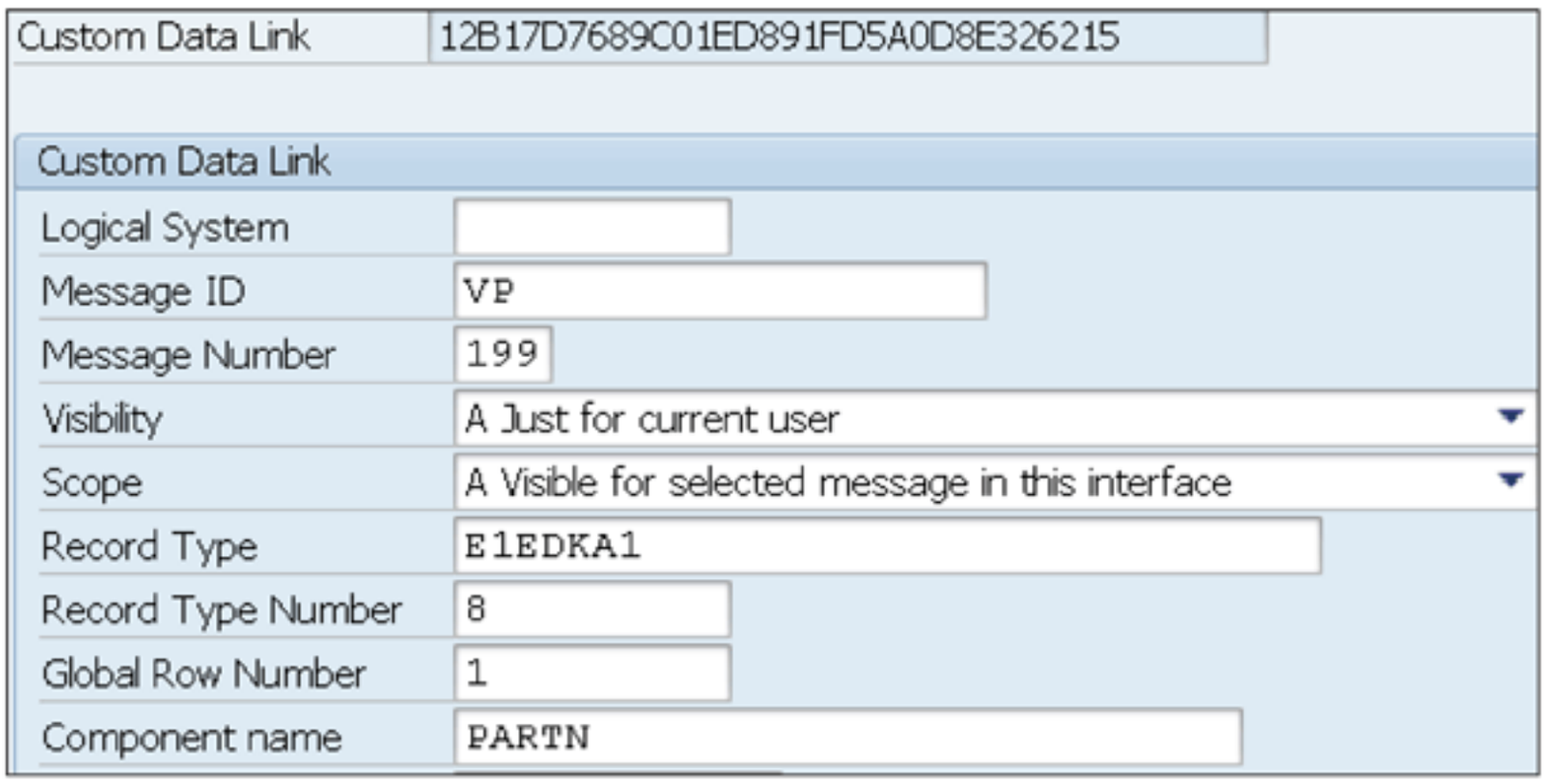

Figure 40 Custom Data Link Maintenance

2.3 SAP Application Interface Framework Indexing

To fully take advantage of monitoring an IDoc interface, you have to create an index table. Index tables store the key values for every single message. Thanks to this, it is possible to identify a message by a specific key in monitoring and error handling. The index table is easy to set up and at the same time speeds up monitoring in SAP Application Interface Framework. Imagine that you receive tons of inbound ORDERS.ORDERS05 messages every single day and you want to find a message by a specific value from the specific field. Because index tables store key values, you can use them to speed up the process of monitoring your interface. In this section, you will be guided step by step through how to create a single-index table and a multi-index table.

Creating Single-Index Tables

To create a new index table for your interface, go to Transaction SE11. There you have to create a new table based on a template table: table `/AIF/STD_IDX_TBL`. Type name of the template table into the **Database Table** field, and then click **Copy** or press CTRL + F5. In the **To Table** field, type the name of

your own custom table that will be used as an index table for your interface, as shown in Figure 41.

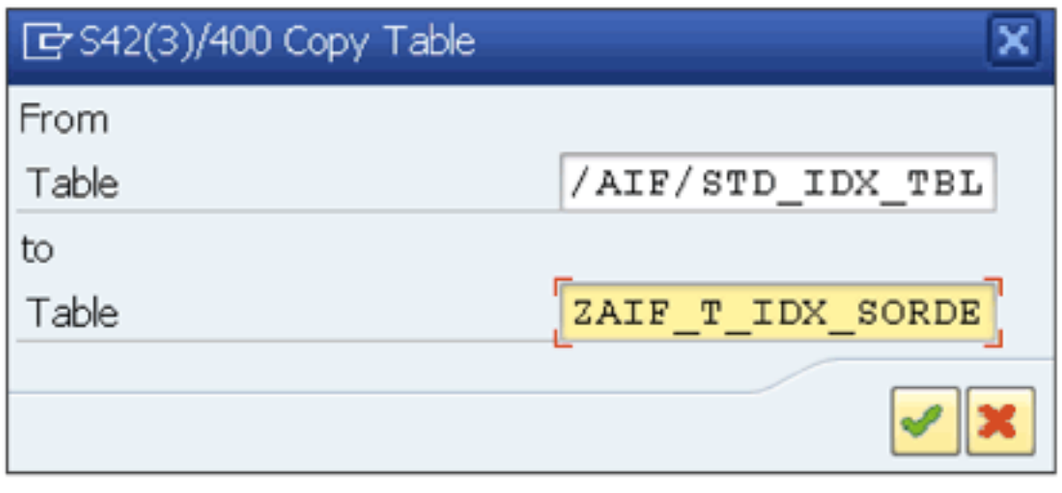

Figure 41 Copy Structure of Table

Once you have created it, open it in edit mode and extend the structure with the key fields for your interface. For the purposes of this E-Bite, we will use **Purchase Order Number** and **Partner Number** as the additional fields in the selection screen. It is worth mentioning that the template table already contains two include structures: `/AIF/IFKEYS`, which consists of the name-space, interface name, and version; and `/AIF/ADMIN`, containing additional administrative information.

Add new **BSTKND** and **KUNNR** fields at the end of the table, as shown in Figure 42.

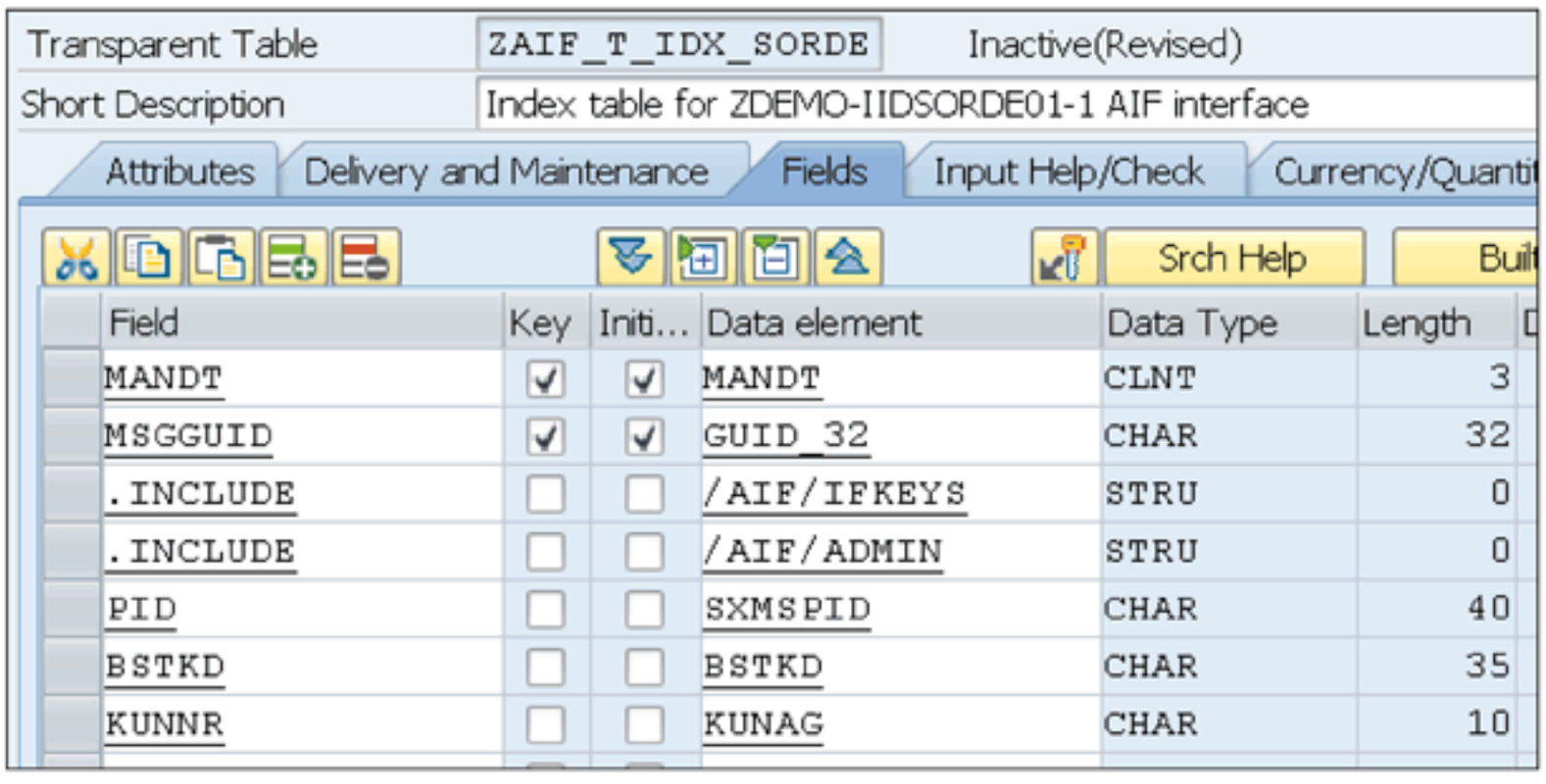

Figure 42 Structure of Single-Index Table

Click the **Save** and **Activate** buttons to activate your new index table. To use your newly created fields, in the selection screen in monitor and error handling, create a new ZAIF_SEL_SCREEN_0001 module pool program type. Go to Transaction SE38 to do so. Make sure that you create the program with **Module Pool** as the **Type**, as shown in Figure 43.

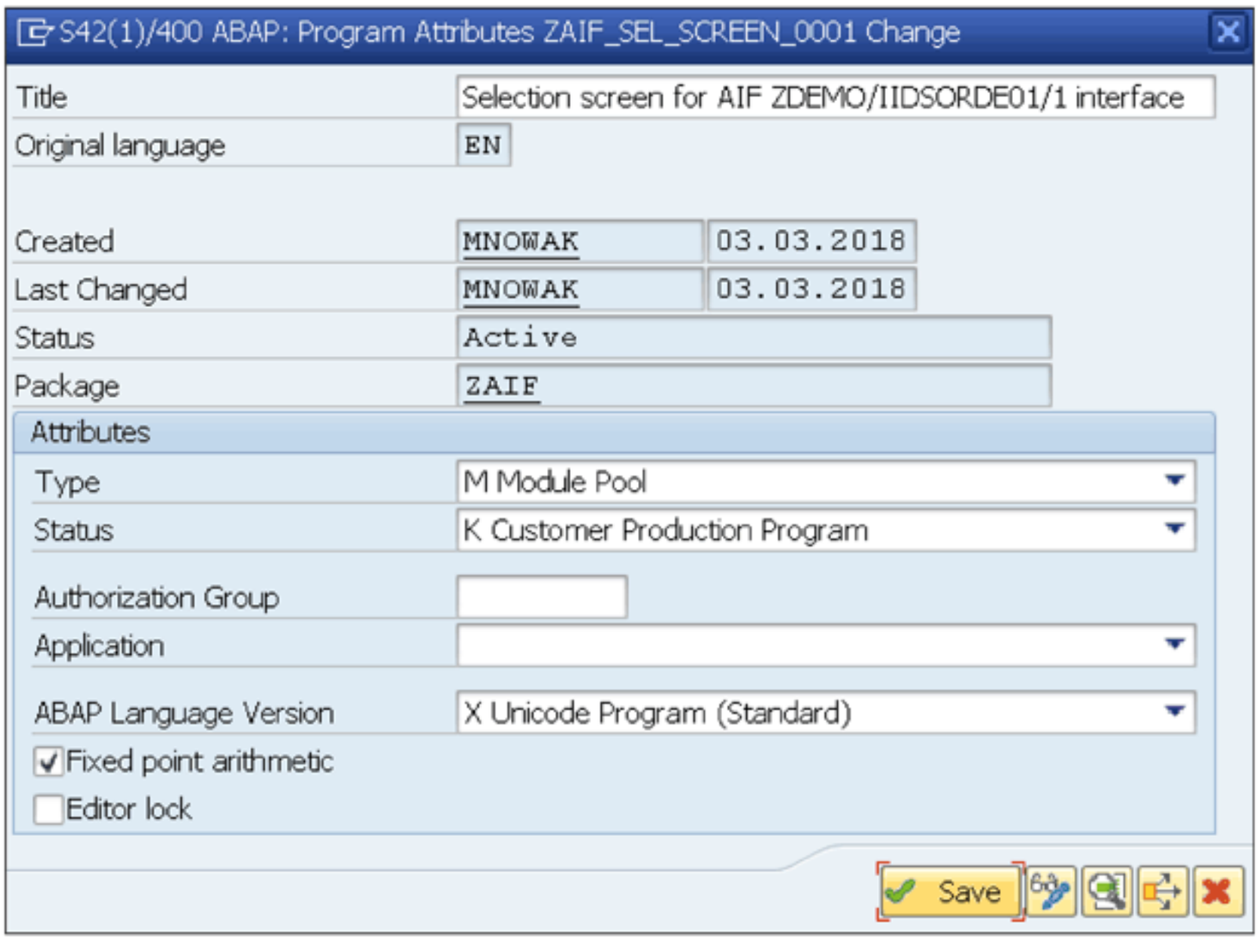

Figure 43 Attributes of Module Pool Program

In this program, it will be necessary to define a subscreen that will be used in the selection screen of the SAP Application Interface Framework monitor. Inside the subscreen, you have to define select options for the sales order number (`s_so`) and purchase order number (`p_so`). At the end, use the `get_value_from_mem` static method of the `/aif/cl_global_tools` class. See the following code snippet for code to use in your module pool.

```
PROGRAM ZAIF_SEL_SCREEN_0001.
DATA: gv_po_num TYPE bstkd,
      gv_kunnr  TYPE kunag.

SELECTION-SCREEN BEGIN OF SCREEN 0001 AS SUBSCREEN.
```

```
SELECT-OPTIONS:
    s_po  FOR gv_po_num,
    s_kun FOR gv_kunnr.
SELECTION-SCREEN END OF SCREEN 0001.

AT SELECTION-SCREEN OUTPUT.
  /aif/cl_global_tools=>get_value_from_mem( ).
```

We strongly recommend also going to **Goto • Text Elements • Selection Text** in your module program to adjust the texts and make them more user-friendly, as shown in Figure 44. After making adjustments, **Save** and **Activate** the program.

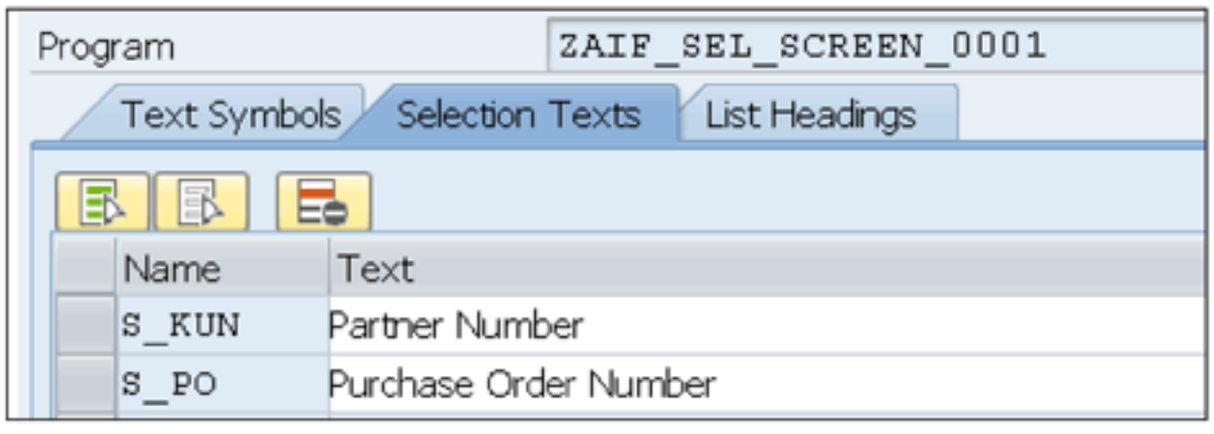

Figure 44 Module Pool Selection Texts

In the last steps, you have to set a single-index table, module pool program, and new fields in SAP Application Interface Framework Customizing. To do so, go to Transaction /N/AIF/CUST and then open **SAP Application Interface Framework • Error Handling • Define Namespace-Specific Features**. Provide your namespace name, and in **Define Interface-Specific Features** click **New Entries**. Here you can provide the following information:

- **Message Idx Table**
  Name of the created single-index table
- **Program Name**
  Module pool program created with subscreens
- **Screen Number**
  Subscreen number defined in the module pool program

Optionally, you can also define the following:

- **Weighting Factor**
  As a number
- **Display Root Field**
  Enables display of the root field
- **Display Description**
  Enables display of the description

Once you fill in this information, as shown in Figure 45, save your changes and assign them to a transport.

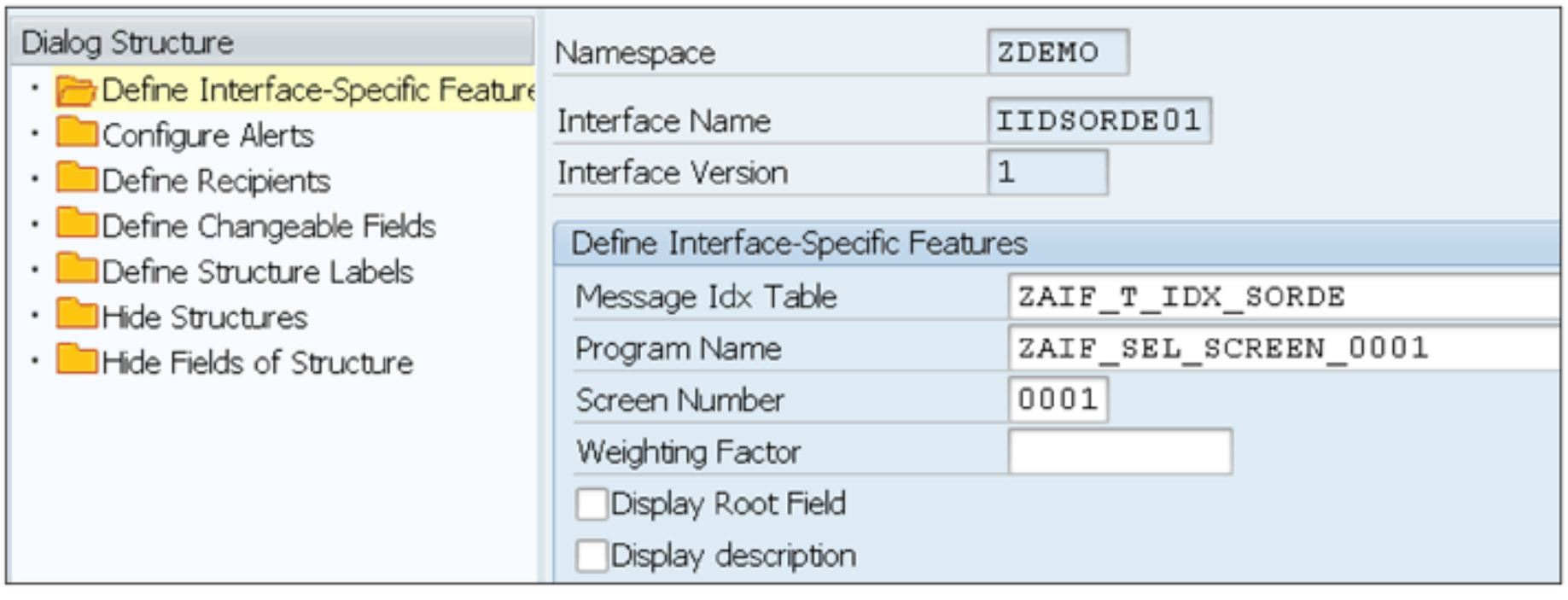

Figure 45 Interface Specific Features

Now you have to assign key fields from the index table in SAP Application Interface Framework Customization. To do so, open Transaction /N/AIF/CUST, then go to **SAP Application Interface Framework • Error Handling • Define Interface-Specific Features**. Here you have to add two entries for each field extended in the index table. You have to provide the following:

- **Field Sequence No.**
  Identifier for each field per interface
- **Key Field Name**
  Name of the field used in the index table
- **Data Element**
  Type of the field used in the index table

- **Name Select-Options/Parameter**
  Name of the select options or parameters used in module pool for this field
- **Field Is Select Option**
  Mark if this field is used as select option in the module pool
- **Field Name**
  Name of the field in SAP structure
- **Raw or SAP Structure**
  Provide **Dest. Structure** because we're using an inbound interface
- **Multi Selection Type**
  Single Selection for single-index tables, **Multi Selection** for multi-index tables, and **Document ID** for document index tables
- **Hide Tree Node in the View 1 Tree**
  Hides the key field name in monitor and error handling

Optionally, you can provide the following:

- **Weighting Factor**
  As a number
- **Parent Field Sequence Number**
  Existing key field sequence number
- **Icon**
  ID of an icon
- **Tooltip**
  Short text displayed in the hierarchy tree as a tooltip
- **Field Name in Alert Recipient Assignment**
  Field used in alert recipient assignment
- **Relevant for Recipient Determination**
  Choose whether the key field is relevant for recipient determination
- **Category Field Name**
  Container element name for namespace field of the alert category

- **Key Field Rule**
  Determines if the key field has a key field rule

For the **Purchase Order** field, fill all fields as shown in Figure 46; for **Partner Number**, provide the information shown in Figure 47.

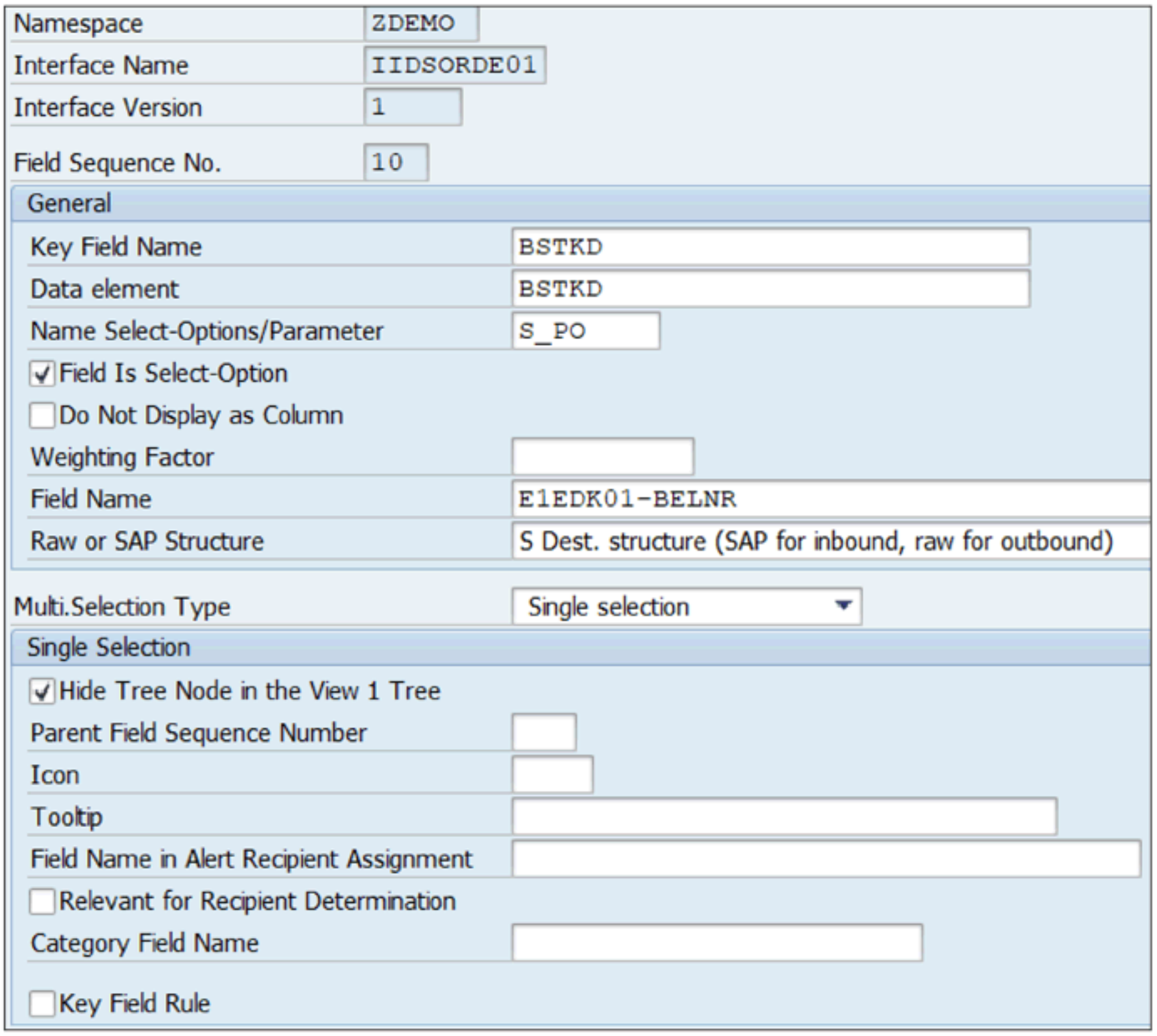

Figure 46 Interface-Specific Features for Purchase Order Number

Congratulations! You have just created your first index table, with two additional fields used in the selection screen. You can go to monitor and error handling and search for messages by specific **Purchase Order Number** and **Partner Number**—for example, purchase order number INT4_10007 and a partner number containting 1010, as shown in Figure 48.

Namespace: ZDEMO
Interface Name: IIDSORDE01
Interface Version: 1
Field Sequence No.: 20

General
- Key Field Name: KUNNR
- Data element: KUNAG
- Name Select-Options/Parameter: S_KUN
- [x] Field Is Select-Option
- [ ] Do Not Display as Column
- Weighting Factor:
- Field Name: E1EDKA1-PARTN
- Raw or SAP Structure: S Dest. structure (SAP for inbound, raw for outbound)

Multi.Selection Type: Single selection

Single Selection
- [x] Hide Tree Node in the View 1 Tree
- Parent Field Sequence Number:
- Icon:
- Tooltip:
- Field Name in Alert Recipient Assignment:
- [ ] Relevant for Recipient Determination
- Category Field Name:
- [ ] Key Field Rule

Figure 47 Interface-Specific Features for Partner Number

Application Selection
- Application: AIF

Application-Specific Selection
- Namespace: ZDEMO
- Interface Name: IIDSORDE01
- Interface Version: 1

Select Interface

- Message Class:
- Message Number:

More Specific Selection
- Purchase Order Number: INT4_10007 to
- Partner Number: *1010* to

Figure 48 Monitor and Error Handling Screen

Click the **Execute** button (or press F8), and you should see only the message that contains the purchase order number and partner number you searched for.

Creating Multi-Index Tables

Single-index tables allow you to create selection criteria for a specific interface for values that appear only once in single message (e.g., purchase order number). If you want to configure multiple selection tables for indexing the values appearing more than once in a single message (e.g., material number), multi-index tables come in handy. In this E-Bite, we will explain how to create multi-index tables for material numbers and plant IDs on the item level. First, create an additional table in Transaction SE11 based on a template table: `/AIF/STD_IDX_TBL`. Create table `ZAIF_T_IDX_SORDM` in the same way as you created single-index table `ZAIF_T_IDX_SORDE` in the previous subsection. Then, you have to add additional fields to the table. It is important to add a **COUNTER** field with the **INT4** type right after the **MSGUID** field. Make sure that you have marked the **Key** and **Initial** checkboxes. At the end of the table, add the **MATNR** field with the **MATNR** type for the material number and the **WERKS** field with the **WERKS_EXT** type for the plant on the item level. Make sure that your table looks like that shown in Figure 49.

Transparent Table ZAIF_T_IDX_SORDM Active

Short Description Multi index tablefor ZDEMO-IIDSORDE01-1 AIF interface

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quant

Srch Help | Bui

Field	Key	Initi...	Data element	Data Type	Length
MANDT	✓	✓	MANDT	CLNT	3
MSGGUID	✓	✓	GUID_32	CHAR	32
COUNTER	✓	✓	INT4	INT4	10
.INCLUDE			/AIF/IFKEYS	STRU	0
.INCLUDE			/AIF/ADMIN	STRU	0
PID			SXMSPID	CHAR	40
MATNR			MATNR	CHAR	40
WERKS			WERKS_EXT	CHAR	4

Figure 49 Structure of Multi-Index Table

Because we will use a multi-index table and single-index table for the same interface, we have to extend the module pool program we created in the previous subsection. Go to Transaction SE38 and open module pool ZAIF_SEL_SCREEN_0001 in edit mode. In this program, you have to extend the selection screen for the **MATNR** and **WERKS** fields according to the final module pool, as shown in here:

```
PROGRAM ZAIF_SEL_SCREEN_0001.
DATA: gv_po_num TYPE bstkd,
      gv_kunnr  TYPE kunag,
      gv_matnr  TYPE matnr,
      gv_werks  TYPE werks_ext.

SELECTION-SCREEN BEGIN OF SCREEN 0001 AS SUBSCREEN.
SELECT-OPTIONS:
    s_po  FOR gv_po_num,
    s_kun FOR gv_kunnr,
    s_mat FOR gv_matnr,
    s_wer FOR gv_werks.
    .
SELECTION-SCREEN END OF SCREEN 0001.

AT SELECTION-SCREEN OUTPUT.
  /aif/cl_global_tools=>get_value_from_mem( ).
```

Remember also to adjust the selections by going to **Menu • GoTo • Text Elements • Selection Text**, as shown in Figure 50. Then **save** and **activate** program.

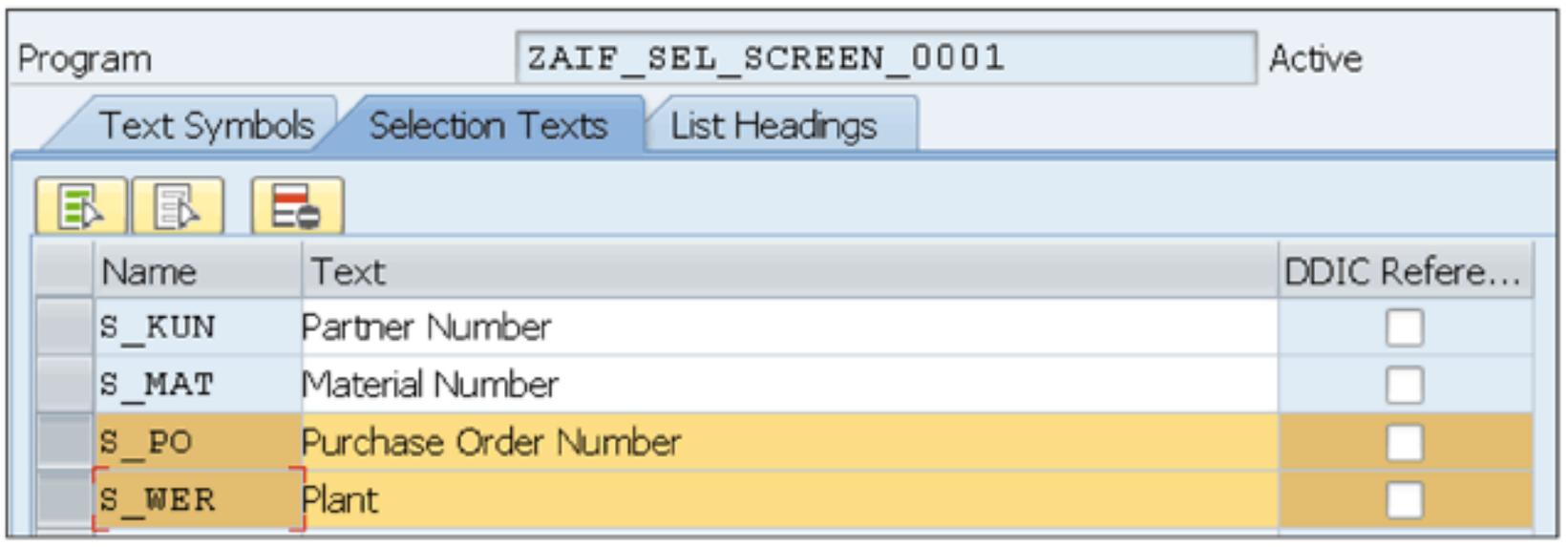

Figure 50 Maintaining Selecton Text of the Report

The final step is to configure the multi-index table in SAP Application Interface Framework Customizing. Open Transaction /N/AIF/CUST and go to **SAP Application Interface Framework • Error Handling • Define Interface-Specific Features** and provide **namespace**, **Interface Name**, and **Interface Version**.

> **Note**
>
> The **Define Namespace-Specific Features** step is only obligatory for single-index tables; you should skip this step for multi-index tables.

Click **New Entries**; in the newly opened window, first change **Multi-Selection Type** to **M Multiple Selection** and press Enter. This will adjust the opened window, and a new **Message Index Table Name** field will appear. In this new field, provide the name of the multi-index table you created: `ZAIF_T_IDX_SORDM`. Then provide the rest of the information for the material number. In the **Key Field Name** field and **Data Element** field, type "MATNR". For **Name Select-Options/Parameter**, enter "S_MAT". Mark the **Field Is Select-Option** checkbox, and in the end change structure to "S Dest. structure (SAP for inbound, raw for outbound)". Now, data will be derived from the message after all mapping (that is not part of this E-Bite). Next, in **Field Name** enter "E1EDP01-E1EDP19-IDTNR". The final entry for material number should look as shown in Figure 51.

Go back and add another entry, this time for plant ID. Select the proper multi-selection type, provide an index table, and then type "WERKS" as the **Key Field Name**, "WERKS_EXT" as the **Data Element**, and "S_WER" as the **Name Select-Options/Parameter**. Again, mark the **Field Is Select-Option** checkbox and then enter "E1EDP01-WERKS" in **Field Name**. The entry for plant ID should be exactly the same as shown in Figure 52.

Namespace	ZDEMO
Interface Name	IIDSORDE01
Interface Version	1
Field Sequence No.	30
General	
Key Field Name	MATNR
Data element	MATNR
Name Select-Options/Parameter	S_MAT
☑ Field Is Select-Option	
☐ Do Not Display as Column	
Weighting Factor	
Field Name	E1EDP01-E1EDP19-IDTNR
Raw or SAP Structure	S Dest. structure (SAP for inbound, raw for outbound)
Multi.Selection Type	M Multiple selection
Multiple Selection	
Message Index Table Name	ZAIF_T_IDX_SORDM

Figure 51 Interface-Specific Features for Material Number

Namespace	ZDEMO
Interface Name	IIDSORDE01
Interface Version	1
Field Sequence No.	40
General	
Key Field Name	WERKS
Data element	WERKS_EXT
Name Select-Options/Parameter	S_WER
☑ Field Is Select-Option	
☐ Do Not Display as Column	
Weighting Factor	
Field Name	E1EDP01-WERKS
Raw or SAP Structure	S Dest. structure (SAP for inbound, raw for outbound)
Multi.Selection Type	M Multiple selection
Multiple Selection	
Message Index Table Name	ZAIF_T_IDX_SORDM

Figure 52 Interface-Specific Features for Plant ID

You have just configured a multi-index table for the inbound sales order interface. Now when a new inbound IDoc arrives in the system, table ZAIF_T_IDX_SORDM will store all entries with the material number and plant ID. To

check this, go to Transaction SE16N and provide the name of your table. Display the entries, and you should see **Number**, **Material**, and **Plant** columns filled as shown in Figure 53. In this figure, you can see that GUID 213006 contains two line, because this message contains two item lines and both of them were saved.

Search in Table: ZAIF_T_IDX_SORDM — Standard index table
Number of hits: 3
Runtime: 0 — Maximum no. of hits

GUID 16	Number	NS	Interface	Version	Material	Plant
0000000000213006	1	ZDEMO	IIDSORDE01	1	MZ-FG-C900	DE10
0000000000213006	2	ZDEMO	IIDSORDE01	1	MZ-RM-M525-04	DE10
0000000000213008	1	ZDEMO	IIDSORDE01	1	MZ-RM-M525-04	DE10

Figure 53 Content of Multi-Index Table

You can also go to Transaction /N/AIF/ERR, and for the specified interface you should see an additional selection screen with extended fields, as shown in Figure 54.

More Specific Selection

Purchase Order Number
Partner Number
Material Number
Plant

Figure 54 Monitor and Error Handling Selection Screen

Advanced Indexing with Key Field Rules

To accommodate more complex indexing requirements, SAP Application Interface Framework provides key field rule customizing. When enabled, it makes more advanced options available to help derive the expected indexing key field value. The additional features are as follows:

- Specify additional key fields to help determine value of current field
- Specify a value mapping that will be executed to determine the value of the current indexing field

- Specify a function module that can execute code necessary to determine the required value for a key field

> **Note**
>
> Key field rules can be used only with single-selection indexing fields. This option is unavailable for multiple-selection indexing fields.

The key field rules are enabled separately for each field using the **Key Field Rule** checkbox from Transaction **/AIF/CUST** (SAP AIF Customizing), after following the path **SAP Application Interface Framework • Error Handling • Define Interface-Specific Features**. You can see the window on Figure 55.

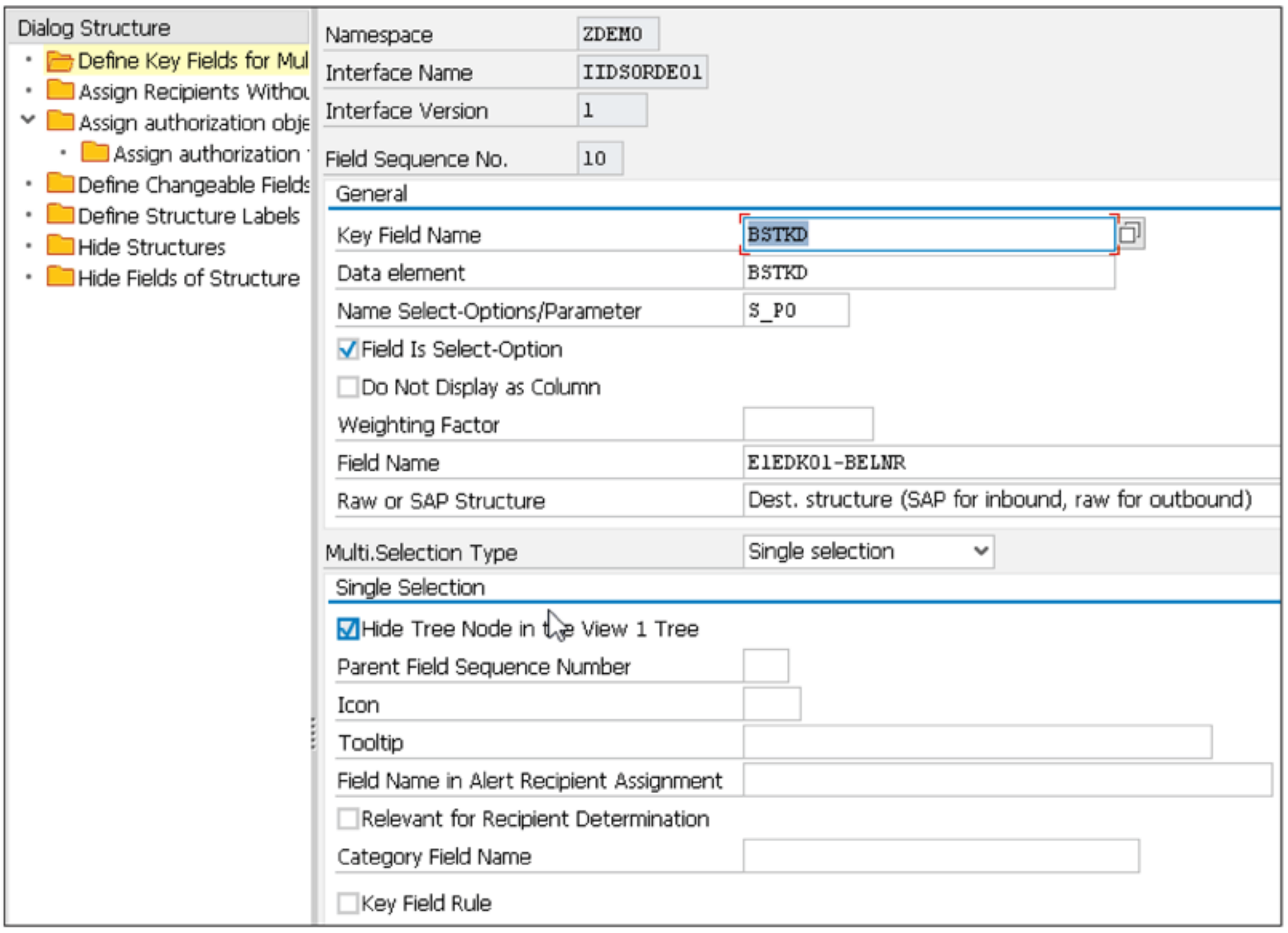

Figure 55 Define Interface-Specific Features: Key Field Rule

Once the **Key Field Rule** checkbox is selected and you press Enter, the screen will expand with additional input fields as shown in Figure 56.

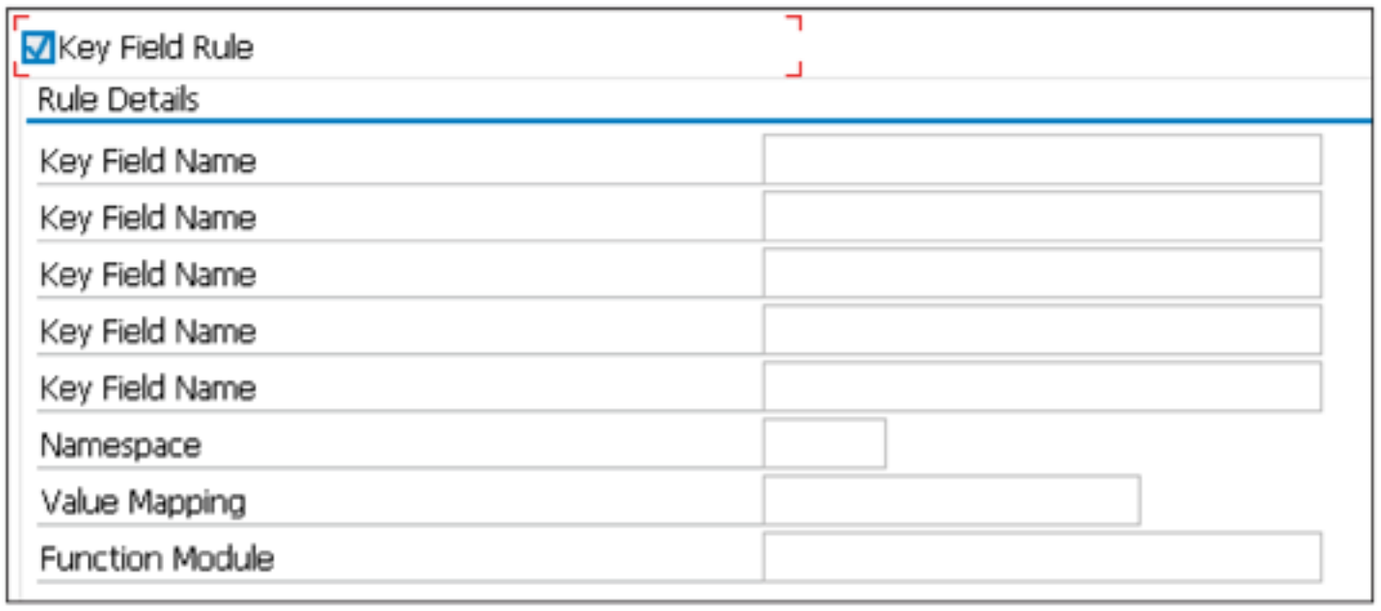

Figure 56 Key Field Rule Input Fields

- **Key Field Name**
  Name of another key field; can be selected from among those defined in the indexing table
- **Namespace**
  Namespace for value mapping object
- **Value Mapping**
  Value mapping object
- **Function Module**
  Template-based function module in which more advanced calculation logic may be applied

As you can see, one option to use the key field rule would be to take advantage of the value mapping object. We could, for example, use the external customer number that we would index in the EXTERNAL_CUSTOMER_NO key field and use a mapping to MAP_EXT_CUSTOMER to derive an internal customer number and store it in a new key field as shown in Figure 57.

As mentioned earlier, entering a custom function module name in the **Function Module** field provides even more capabilities. The function module should be based on pre-delivered template /AIF/TEMPL_ENH_KFLDS (as shown in Figure 58). It has access to both raw and SAP message structures as

importing parameters. The changing CS_SIDX_ENTRY structure can be used to update the content of the key fields. The IS_KFLDS structure provides context for the function call—for example, the name of the key field currently processed and key field customizing setup.

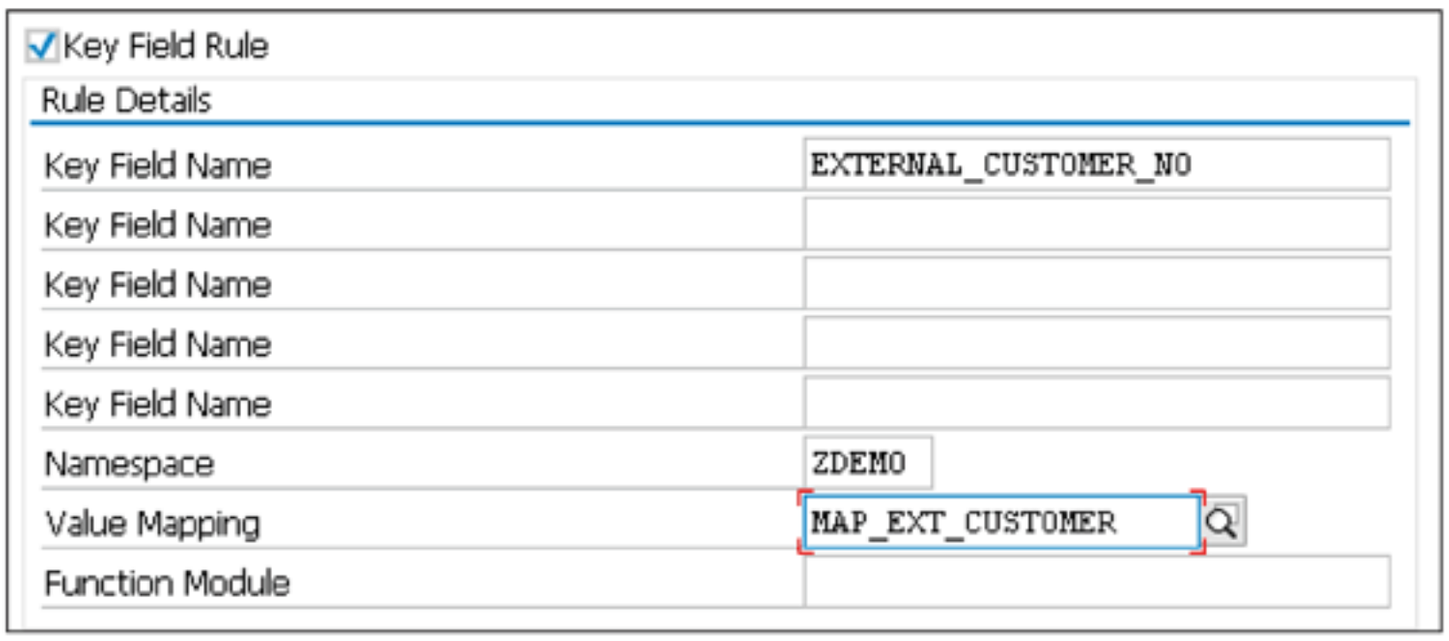

Figure 57 Key Field Rule with Use of Value Mapping

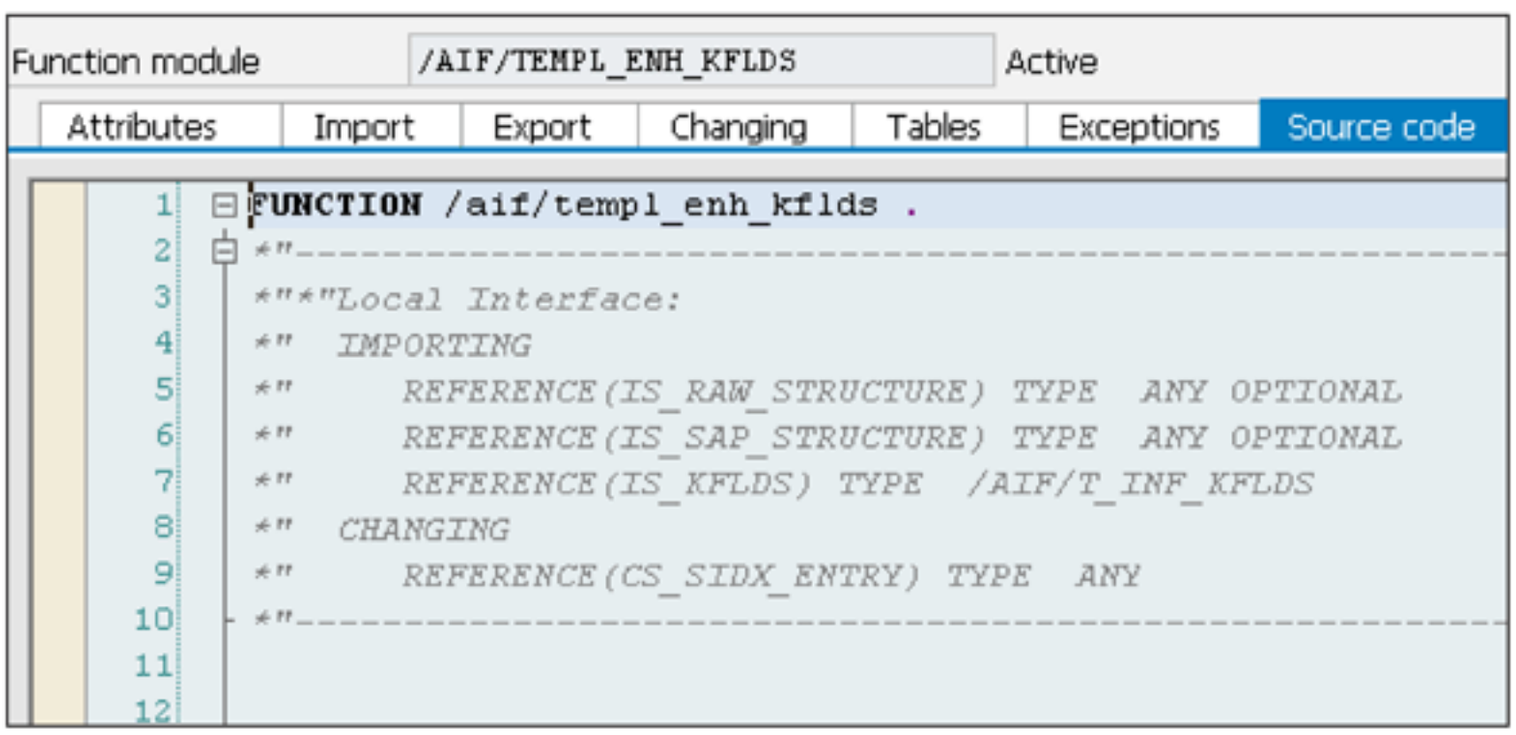

Figure 58 Key Field Template Function Module

Let's run through a simple example in which we take advantage of the key field rule function module to set the value of a key field. We create a copy of /AIF/TEMPL_ENH_KFLDS named ZAIF_ENH_KFLDS_DEMO and enter its name into the **Function Module** field. We will also reference key field KUNNR, so we enter it in the **Key Field Name** field. Example is shown in Figure 59.

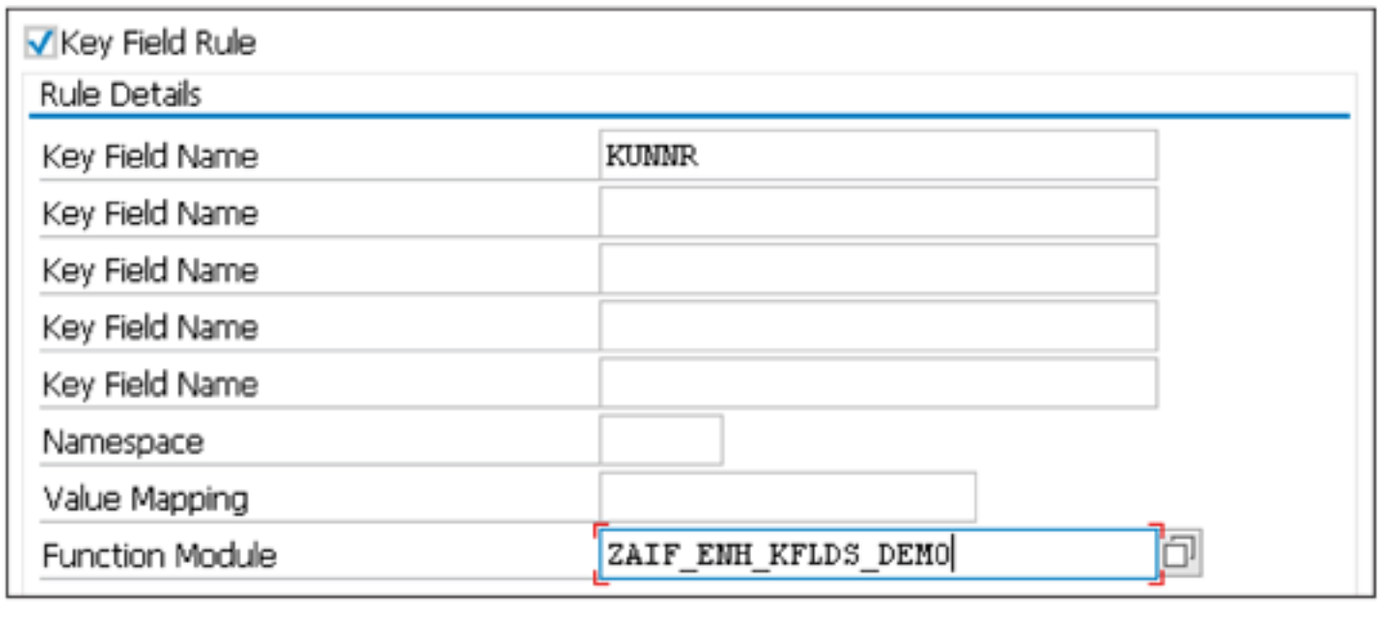

Figure 59 Key Field Rule: Function Module Assignment

The following demonstrates simple code that reads the value of the key field parameter (`IS_KFLDS-FIELDNAME1`): `KUNNR`. Then it concatenates it with the prefix `XY` and populates the currently processed key field (`IS_KFLDS-FIELDNAME`) with the resulting value.

```
FUNCTION ZAIF_ENH_KFLDS_DEMO .
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(IS_RAW_STRUCTURE) TYPE  ANY OPTIONAL
*"     REFERENCE(IS_SAP_STRUCTURE) TYPE  ANY OPTIONAL
*"     REFERENCE(IS_KFLDS) TYPE  /AIF/T_INF_KFLDS
*"  CHANGING
*"     REFERENCE(CS_SIDX_ENTRY) TYPE  ANY
*"----------------------------------------------------------------------
  " Assign value of key field from 1st parameter
  ASSIGN COMPONENT is_kflds-fieldname1
    OF STRUCTURE cs_sidx_entry TO FIELD-SYMBOL(<fs_kunnr>).
  " Assign value of currently processed key field
  ASSIGN COMPONENT is_kflds-fieldname
    OF STRUCTURE cs_sidx_entry TO FIELD-SYMBOL(<fs_bstkd>).
  IF <fs_kunnr> IS ASSIGNED
    AND <fs_bstkd> IS ASSIGNED.
      <fs_bstkd> = |XY{ <fs_kunnr> }| .
  ENDIF.
ENDFUNCTION.
```

When posting a new message, we can see the result in monitoring and error handling. The key field for **Customer Reference** is the expected concatenation of XY and the **Sold-to Party** field as shown in Figure 60.

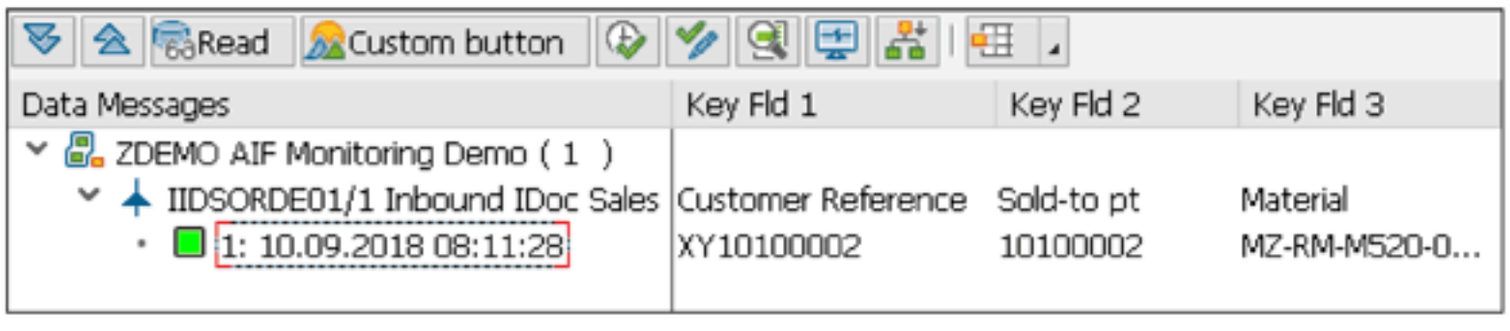

Figure 60 Monitoring and Error Handling: Indexing Fields

2.4 Message Structure Browsing

Thanks to SAP Application Interface Framework's indexing feature, it is easy to find interface messages that contain a value in a specific field. What can we also do to facilitate searching for information inside of messages is utilize some of the message-structure-related functions of the SAP Application Interface Framework monitor? In this section, we will explain how you can rename the structures displayed in the monitor, as well as hide those unnecessary from display, and make browsing the message structure much easier for users.

Custom Labels

Message structures are always very technical, and their components tend to have technical names. That makes it difficult to browse through them to find specific information. Fortunately, in Transaction /AIF/ERR, we have control over the display of the message structure.

The **Define Structure Labels** function allows you to rename selected components so that monitoring users without technical knowledge can easily find the desired information inside of a message. This function is available on two levels: namespace-specific and interface-specific. The difference between the two is that when you rename a certain component in a specific structure using the *namespace-specific* level, the custom label will be visible

for all interfaces that are using the same structure. When you do this on the *interface-specific* level, the change will be valid only for that particular interface. This can be handy when you are using a common message structure, wherein the same component can carry a different information in different scenario.

In this exercise, we will make a change on the interface-specific level, but keep in mind that the process is the same for the namespace-specific level.

Execute Transaction /AIF/CUST and go to the following node: **SAP Application Interface Framework • Error Handling • Define Interface-Specific Features**. Specify your **Namespace**, **Interface Name**, and **Interface Version**, then go to error handling customizing for this interface. On the next screen, double-click the **Define Structure Labels** node and click **New Entries** to start creating a custom structure label. Use search help on the **Component Name** field to select a structure component the label of which you will rename. Select **E1EDK01** as the component. Also, check the **Use DDIC Descr.** option, as shown in Figure 61. Thanks to this option, there is no need to define your own text for the labels. Instead, SAP Application Interface Framework will inherit the descriptions from the data element behind the selected component.

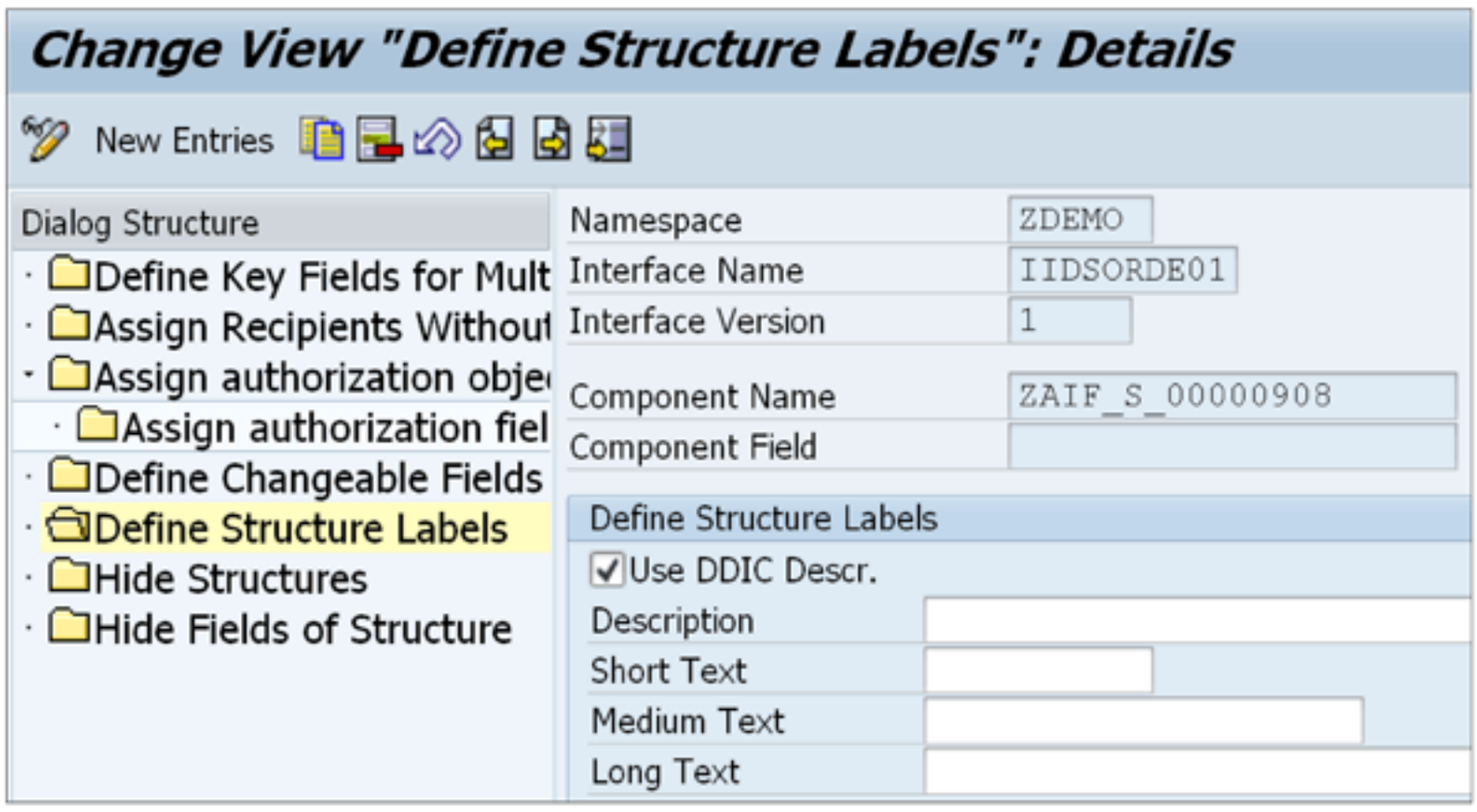

Figure 61 Define Structure Labels: Renaming Compnent Label

Now make another entry, and this time select the same **Component Name**, **E1EDK01**, and additionally specify "ZTERM" as the **Component Field**.

In addition, maintain the rest of the fields as shown in Figure 62.

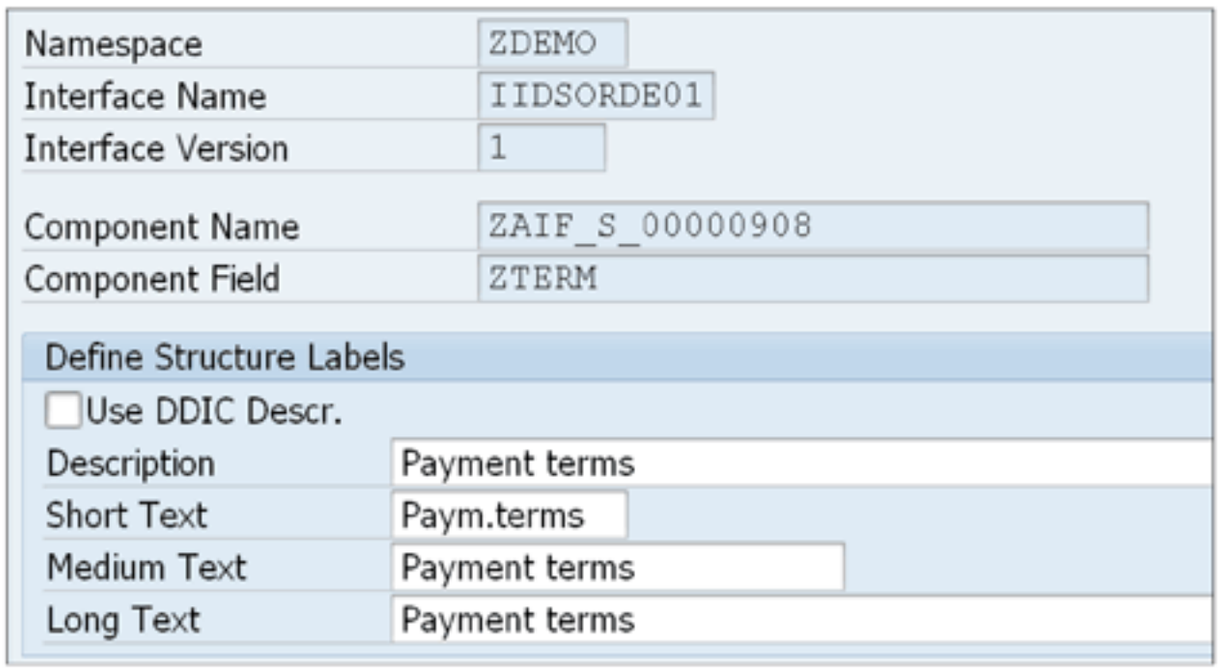

Figure 62 Define Structure Labels: Renaming Field Label

Save your entries and go to the monitoring and error handling transaction, Transaction /AIF/ERR, to test your changes.

On the selection screen, select your interface, and execute the selection. Double-click any message found and check the name of component, **E1EDK01**, and the label for field **E1EDK01-ZTERM**.

You should see a result like the one in Figure 63.

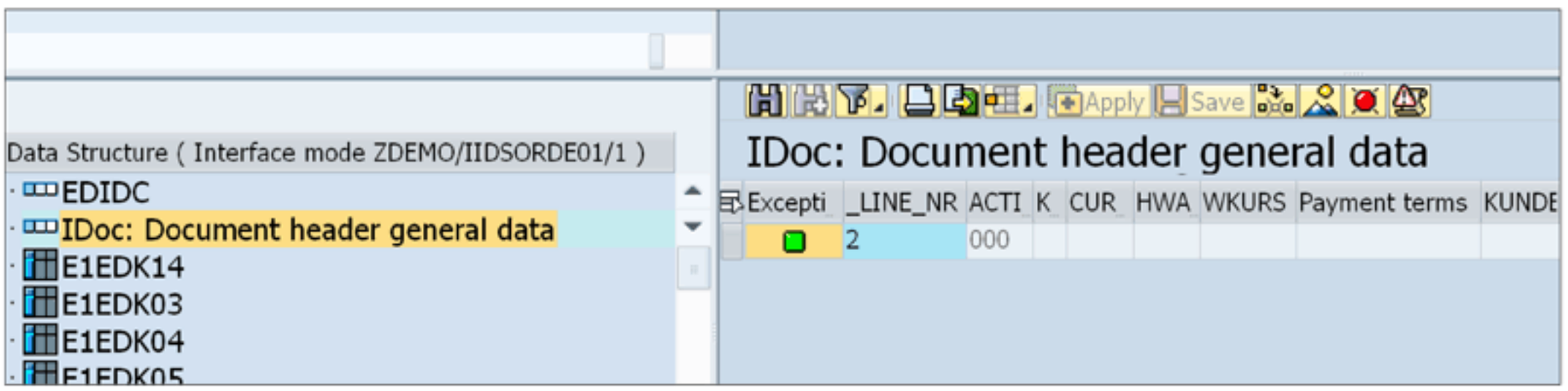

Figure 63 Changed Labels in Transaction /AIF/ERR

Hidden Structure Components

Some components might carry sensitive information that should not be presented to users. Some components might never be filled with data. In any case, we can hide them from display, using SAP Application Interface Framework customizing. Like renaming labels, you can hide structure components on a namespace-specific or interface-specific level. This time, again, we will choose the interface-specific option.

Execute Transaction /AIF/CUST and go to the **SAP Application Interface Framework • Error Handling • Define Interface-Specific Features** node. Specify the name of your interface, and on the next screen double-click on the **Hide Structures** node. Make a new entry and use search help for the **Structure** field. In the next window, select component **E1EDK04** and **Save** your entry. Your entry should look as in Figure 64.

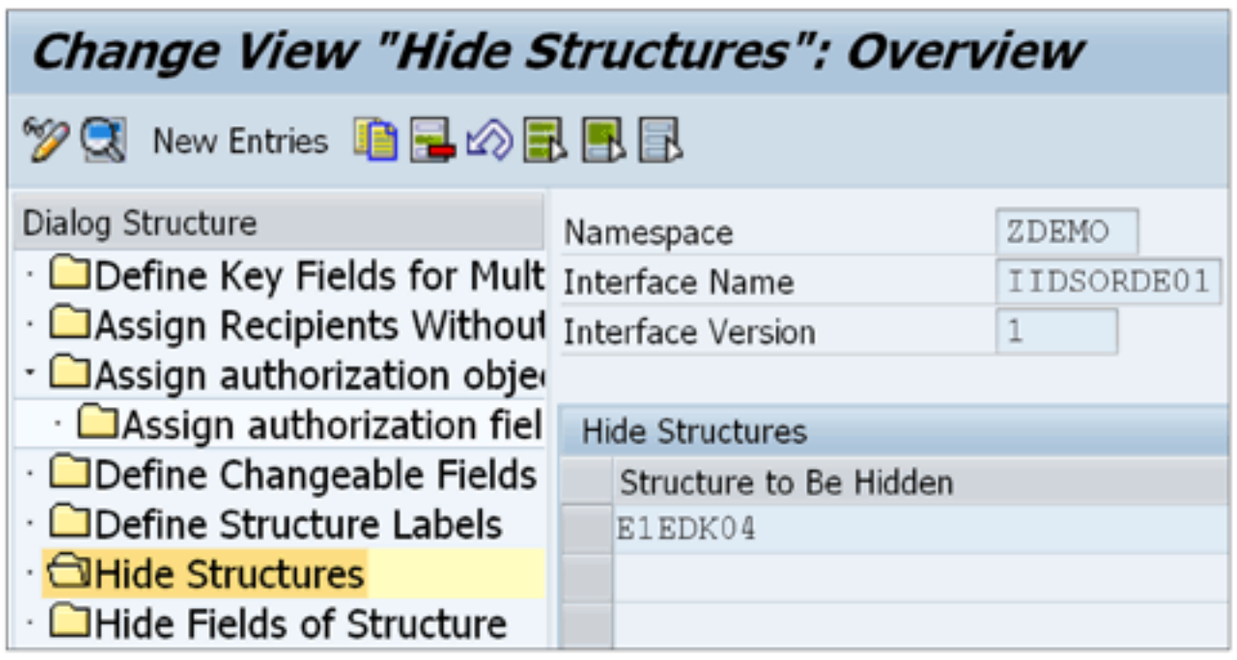

Figure 64 Hiding Components: Selection of Structure

In the next step, go to the **Hide Fields of Structure** node to hide a specific field only. Click **New Entries** and specify a value of "10" as the **Index**. Use search help to select a field for the **Field Path**. Double-click component **E1EDK01** in the left-hand window to get a list of all fields in this component in the right-hand window. Select the **BSART** field, as shown in Figure 65, and click **OK** to confirm.

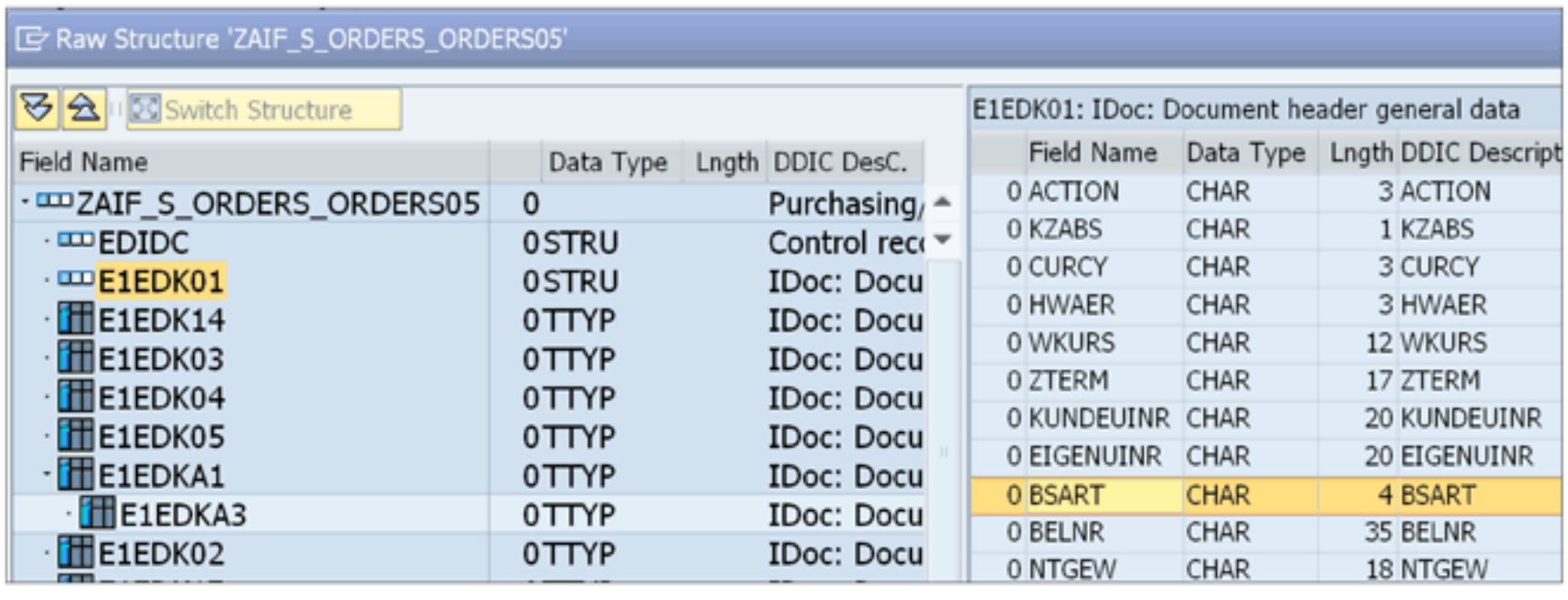

Figure 65 Hiding Fields of Structures

Save your entry and go to Transaction /AIF/ERR to test the changes. Again, select your interface on the selection screen and click **Execute**. Select any message from the list by double-clicking it in the top-left window. Then browse through the structure in the bottom-left window and check if the component **E1EDK04** is hidden now. In addition, double-click the **IDoc: Document Header General Data** component to see if the **BSART** field is also hidden. You should see the result shown in Figure 66.

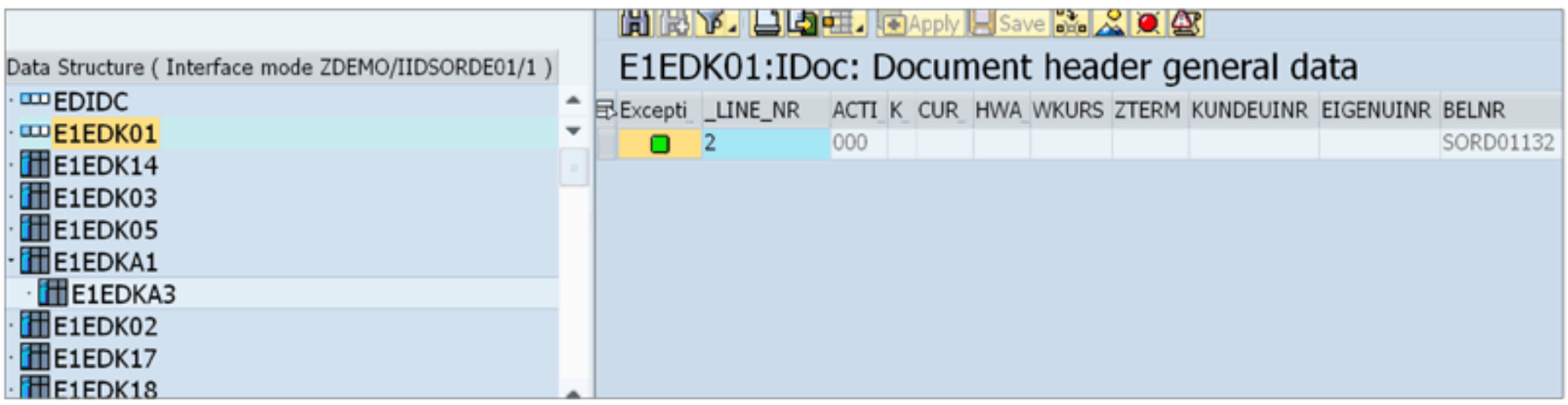

Figure 66 Hidden Structure Component E1EDK04 and Field E1EDK01-BSART

Editable Fields

One of the powerful functionalities of the monitoring and error handling transaction is the ability to change the content of messages directly from the monitor. This, of course, needs to be set up at first and can be done on both the namespace-specific and interface-specific levels. Each field that

will be available for editing from the monitor needs to be specified in customizing. In addition, users who should be able to make such changes in messages need to be authorized in SAP Application Interface Framework, with authorization role /AIF/ERRHDL_CHANGE. Because changing message data is a responsible task, the change authorization should be given only to limited number of users.

Now let's set a field as editable.

1. Execute Transaction /AIF/CUST and go to path **SAP Application Interface Framework • Error Handling • Define Namespace-Specific Features**.

Note

The same can be achieved also on the interface-specific level. This time we will make a change on the namespace-specific level, which means that it will be valid for all interfaces with the given field.

2. Specify your namespace and go to the **Define Changeable Fields** node. Make a **New Entry**, and maintain the following values for the fields listed below:
   - **Index**
     Set to **10**
   - **Field Path**
     Set to **E1EDK01-BELNR** (you can use search help to browse through a specific structure)
3. **Save** your changes and go to Transaction /AIF/ERR to test it.
   Changes are only possible for messages in an error status, so select such a message for your interface, and double-click component **E1EDK01**. You should notice that the **BELNR** field is marked with a darker color, as shown in Figure 67. This is how SAP Application Interface Framework informs the user that changes are possible for this field.

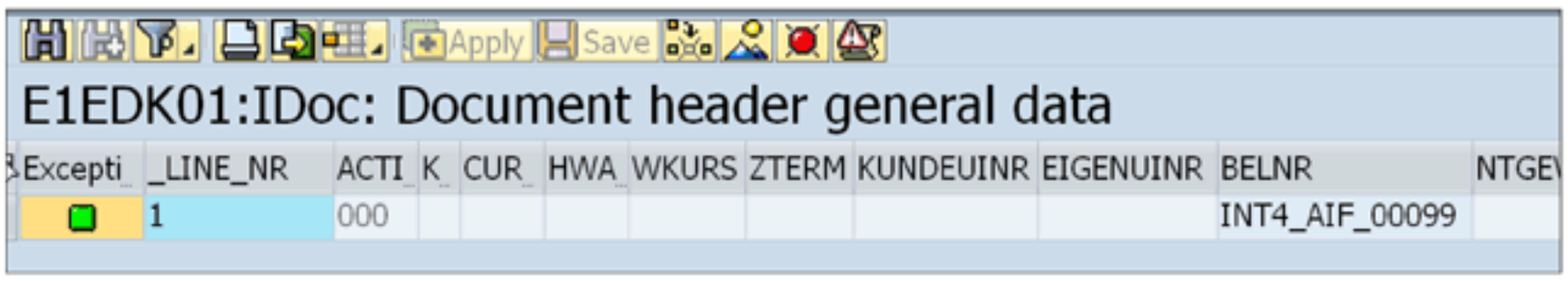

Figure 67 Editable Field in Monitor

4. Now, double-click this field. A pop-up will be shown in which you can change a value to a different one. Change the value of the field to "TEST1" and confirm. You will see that the **BELNR** field will now turn yellow.
5. Now click the **Apply** and **Save** buttons visible in Figure 68.

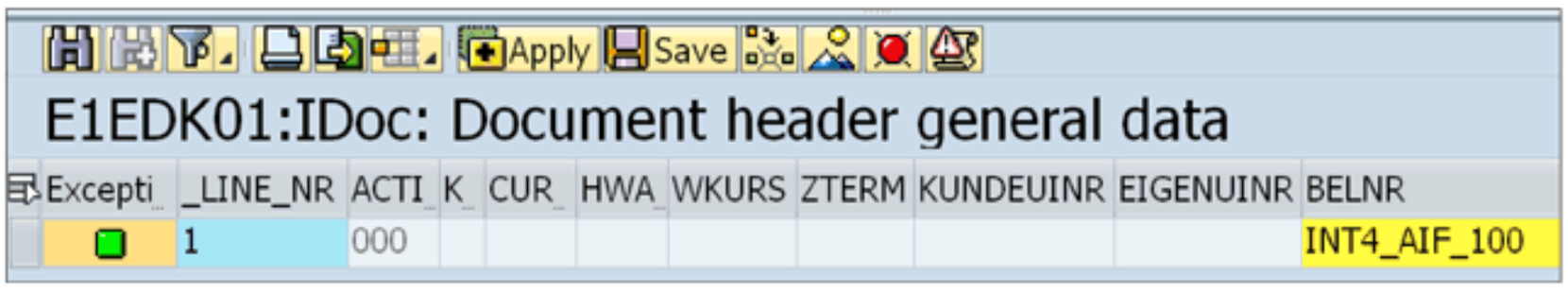

Figure 68 Field in Message after Editing

The status of the message will change to edited, which will be visible in the top-left window, as in Figure 69.

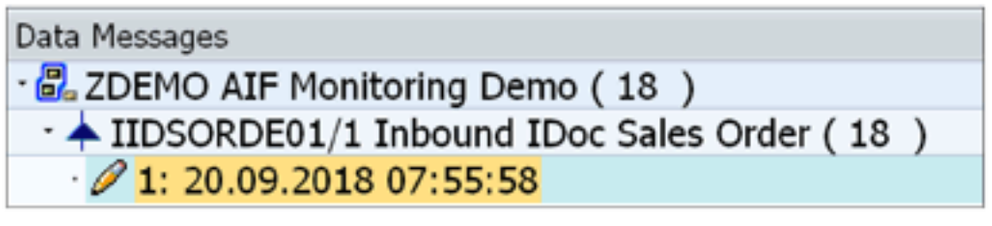

Figure 69 Status of Message after Editing

SAP Application Interface Framework also keeps track of any changes made within message data. A special audit log is available at Transaction /AIF/EDCHANGES. When you execute this transaction, you can use the selection screen to select your interface. In addition, you can specify the following selection parameters:

- **From (Date/Time)**
  Start date and time for the selection
- **To (Date/Time)**
  End date and time for the selection

- **Single Message ID**
  Useful if you are interested in a specific interface message and you are familiar with its ID (e.g., IDoc number)

Execute the report, using your interface name and leaving the rest of the parameters with default values. Double-click your interface in the left-hand window and you should see an entry in the audit log in the right-hand entry. The log of changes, as shown in Figure 70, presents you with the following information:

- **Message ID**
  ID of the changed message
- **NS**
  Namespace of an interface
- **Interface**
  Interface name
- **Version**
  Version of an interface
- **Field Path**
  Path to a changed field
- **Line No.**
  Line number in the segment that was changed
- **New Value**
  New value set by user
- **Old Value**
  Old value before the change
- **Modified On**
  Date of modification
- **Mod Time**
  Time of modification
- **Modified By**
  User who modified the message

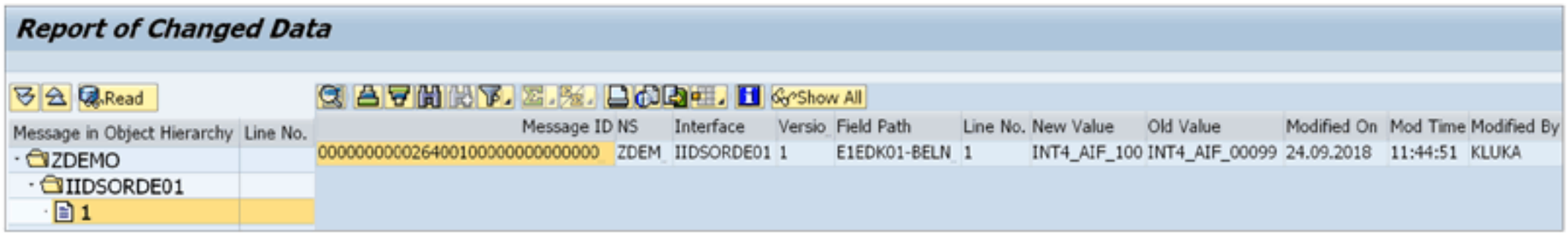

Figure 70 Change Log Transaction

As you can see, thanks to this change log, all changes to message data are transparent and nothing can be changed without a trace being left in the logs.

Dissolve Structures

There is an additional feature that can be used to influence how message data is displayed in monitoring and error handling: dissolve structures. As the name suggests, we can *dissolve* fields into a structure of a higher level. As a result, the dissolved structure will no longer be visible, but its fields will be displayed together with the fields of a parent structure. The aim of this feature is to simplify the hierarchy and reduce nesting levels to provide more information at a glance.

> **Note**
>
> Only child structures can be dissolved into their parents. If you attempt to apply dissolve setting to a table, it will be hidden from the hierarchy instead.

The customizing can be accessed from Transaction /AIF/CUST at the path **SAP Application Interface Framework • Error Handling • Define Global Features** as shown in Figure 71.

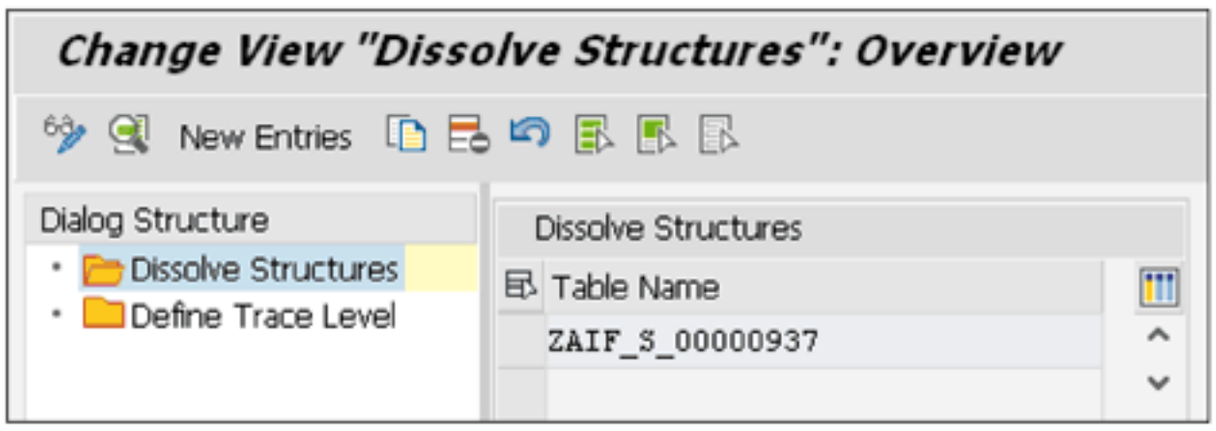

Figure 71 Structures Customizing

To illustrate this feature, let's dissolve IDoc ORDERS05 segment **E1CUREF**, which is nested under the **E1EDP01** segment (as shown in the displayed hierarchy in Figure 72).

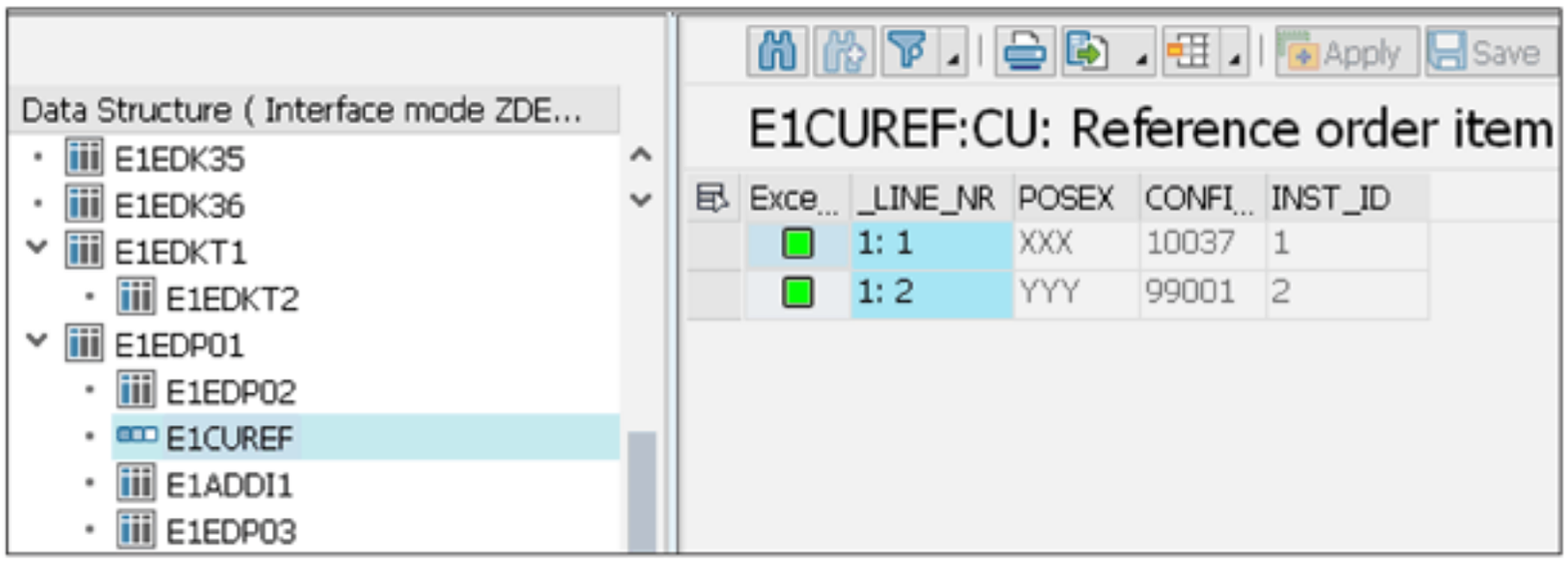

Figure 72 Message Structure before Dissolving

Initially, the content of the E1CUREF segment is displayed separately–see Figure 72. Then the ZAIF_S_00000937 structure type (the generated type for the E1CUREF segment) is entered into the dissolve structures customizing like in Figure 72. Afterwards, as shown in Figure 73, the **E1CUREF** segment is no longer visible under the **E1EDP01** hierarchy.

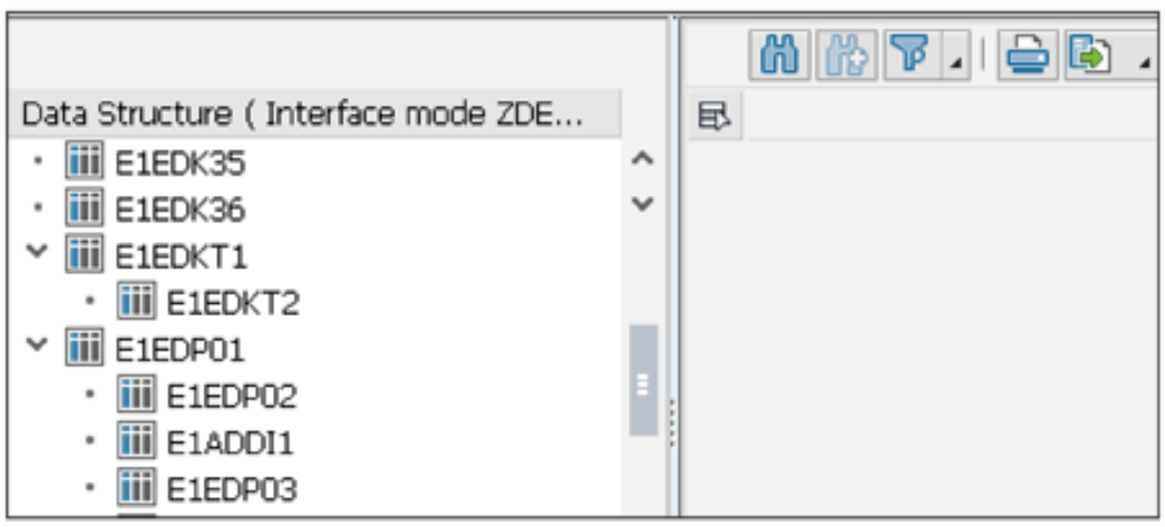

Figure 73 Message Structure after Dissolving

Instead, we will find the fields of **E1CUREF** to be displayed as part of its parent structure—that is, **E1EDP01**. The fields were appended at the very end of **E1EDP01** as shown in Figure 74. Their column names refer now to the segment of their origin, **E1CUREF**.

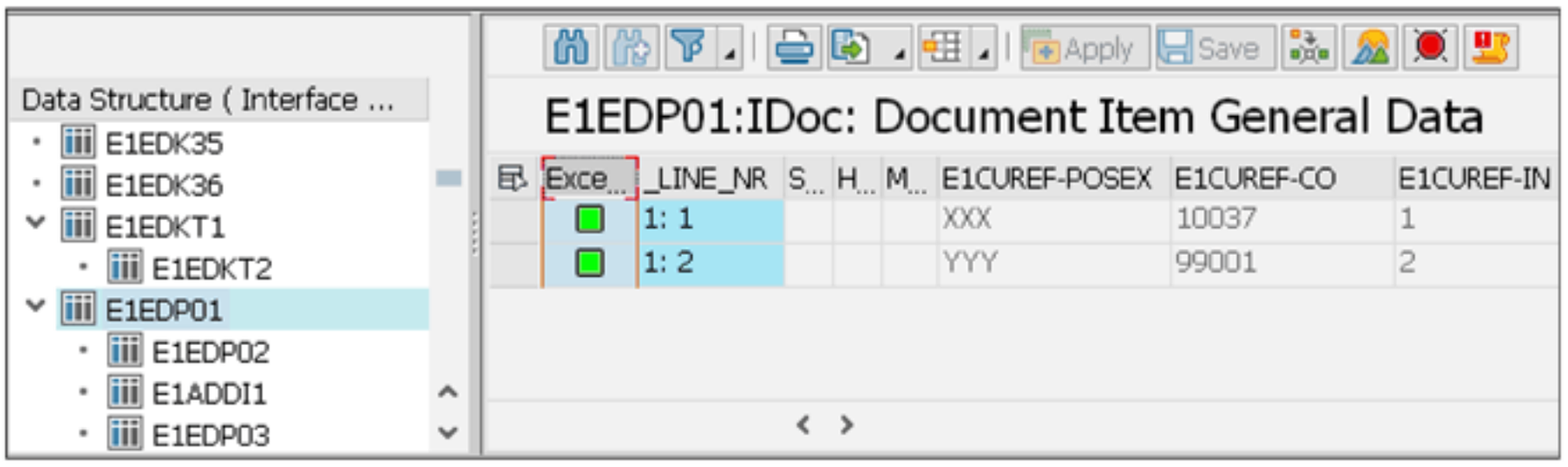

Figure 74 Message Structure after Dissolving: Parent Structure View

2.5 Customizing the Monitoring and Error Handling Transaction

As powerful as the standard SAP Application Interface Framework monitoring and error handling transaction is, there might be situations when you need to extend it to fit your needs. Using enhancement spot `/AIF/ERROR_HANDLING`, you can extend the transaction by adding new buttons, changing the behavior of the existing ones, or making other changes to the standard functionalities, like programmatically controlling the changeable fields in messages. Table 1 lists all the BAdIs that are part of this enhancement spot.

BAdI name	Description
`/AIF/MSG_TXT_MODIFY`	Modify Message Text
`/AIF/SAME_STATUS_AS_ON_PI_BADI`	Set the same status in SAP Application Interface Framework as on PI
`/AIF/V1_ACT`	View 1: Actions
`/AIF/V2_STRUCTURE`	Enhancement for View 2
`/AIF/V3_ACT`	View 3: Actions
`/AIF/V5_ACT`	View 5: Actions
`/AIF/V5_APPL_LOG`	View 5: Enhance Application Log loading
`/AIF/V5_CHANGEABLE_FIELDS`	View 3: Define additional changeable fields
`/AIF/V5_GENERIC_TICKET`	Generic Ticket system for View 5

Table 1 Monitoring and Error Handling BAdIs

In this section, we will focus on the most common enhancement, which is adding a custom button to the monitor. There are alternative ways of performing that action: using a BAdI or using a custom class. Using the BAdI approach, you only change a specific part of the screen and its functionality. Using the custom class, you have control over more functionalities, but it requires more development work. The subsections ahead cover each approach.

The main screen of Transaction /AIF/ERR is split into four or five windows, depending on the mode in which you are using the monitoring and error handling transaction (standard or technical). Each window is referred to as a view in the technical implementation of SAP Application Interface Framework. Table 2 lists identifiers for all windows, which should facilitate searching for a proper BAdI/method when enhancing the standard monitor.

Screen Window	Identifier
Top-left	View 1
Top-middle (visible only in technical mode)	View 4
Top-right	View 5
Bottom-left	View 2
Bottom-right	View 3

Table 2 Window Identifiers in Transaction /AIF/ERR

BAdI Implementation

Adding a new button to a standard SAP screen may sound like tedious work, but in fact it isn't. Using the BAdIs listed in Table 1, you can enhance the transaction with your custom buttons very quickly. In this example, we will add a custom button to the top-left window of the /AIF/ERR monitor, where we see a list of all selected interface messages. Our button, when clicked, will detect the interface details of a selected message and will display an appropriate message to the user.

Start with creating a BAdI implementation. Go to Transaction SE19, and in the **Create Implementation** section, select **New BAdI**. Maintain **Enhancement Spot** as "/AIF/ERROR_HANDLING" and click **Create**.

Maintain the fields in the next pop-up as follows:

- **Enhancement Implementation**
  "ZAIF_ERROR_HANDLING"
- **Short Text**
  "AIF Error Handling Enhancement demo"
- **Composite Enhancement Implementation**
  Leave blank

In the next pop-up, you will be prompted to specify the **BAdI Implementation**, **Implementation Class**, and **BAdI Definition**. Because we are enhancing the top-left window, with identifier `View1` (according to Table 2), we will use BAdI definition `/AIF/V1_ACT`. Populate the fields as shown in Figure 75.

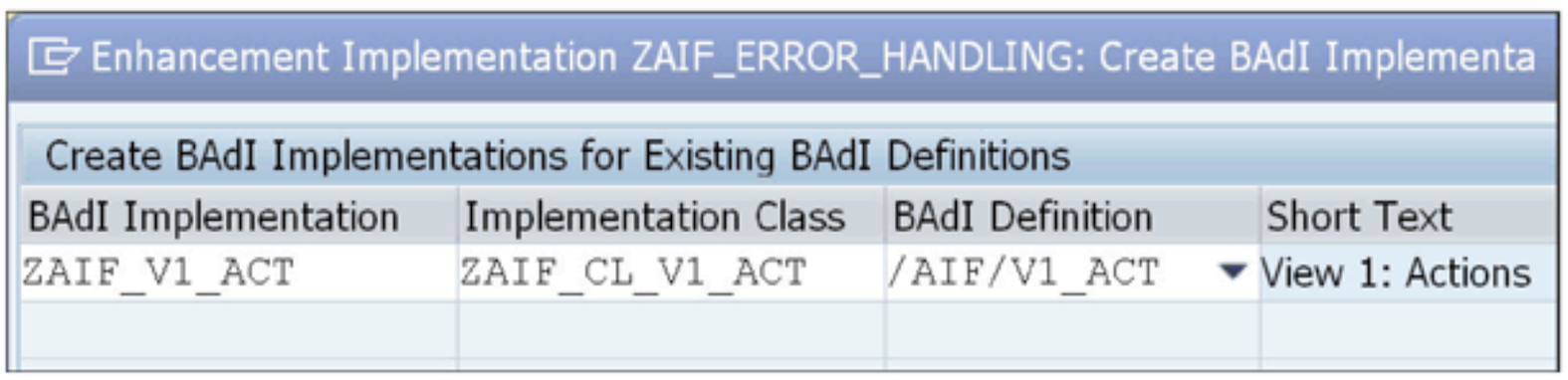

Figure 75 View1 BAdI Implementation

In the next screen(s), you might be asked either to copy/inherit **From Sample Implementation Class**, or just to **Create Empty Implementation**. Choose the latter. When in the **Enhancement Implementation** window, double-click **Implementing Class**, then, in the right-hand window, double-click the **/AIF/IF_V1_ACT~GET_FUNC_LIST** method. Confirm with **Yes** when asked if you want to create a method implementation.

Inside of the method, paste the code snippet from Listing 1.

```
DATA: ls_func TYPE /aif/t_func_lst_st.
ls_func-application_id = 'AIF'.
ls_func-func_code      = 'CUSTBUTTON'.
```

```
ls_func-view_id        = '1'.
ls_func-seqno          = '001'.
ls_func-btn_icon       = '@3Q@'.
ls_func-description    = 'Custom button'.
ls_func-tooltip        = 'Custom Button'.
ls_func-disabled       = ' '.
INSERT ls_func INTO TABLE ct_func_list.
```

Listing 1 Code Snippet for Custom Buttion in View1

Save and **Activate** your code. The preceding method only adds a button to the screen. To attach a real action to the button, you need one more method implementation. Go back to your BAdI implementation, `ZAIF_V1_ACT`, and this time double-click the **/AIF/IF_V1_ACT~DO_ACTION** method. This method will be executed every time a button from `View1` is clicked in the monitor. Paste the code snippet from Listing 2.

```
IF iv_func_code = 'CUSTBUTTON'.
  MESSAGE i000(/aif/error_handling) WITH
         'You selected message of interface '
         is_aif_key-ns is_aif_key-ifname is_aif_key-ifver.
ENDIF.
```

Listing 2 Action Attached to a Custom Button

Remember to **Save** and **Activate** your changes.

Now go to Transaction /AIF/ERR and see if the new button is visible. Select your test interface and execute the selection. You should see your new button on top of the message list window (View1), as shown in Figure 76.

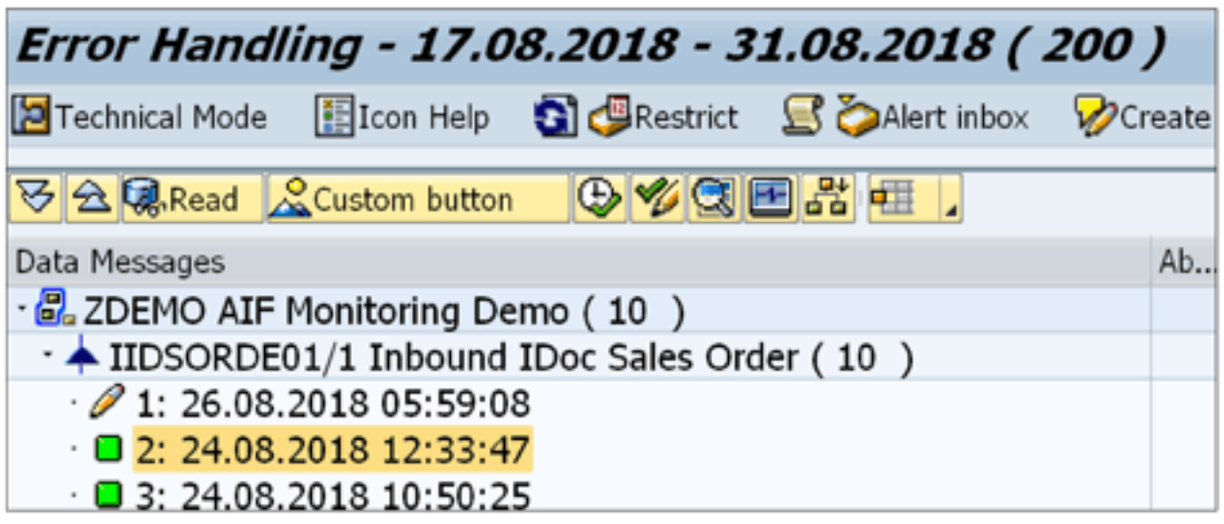

Figure 76 Custom Button Added to Transaction /AIF/ERR

Now, select one of your messages and click the **Custom** button. You should see a message like that in Figure 77.

Figure 77 Message in Response to Clicking Custom Button

This completes our example of using BAdIs to enhance Transaction /AIF/ERR.

Custom Class

We have also an alternative way of enhancing the SAP Application Interface Framework monitor using a custom action handler class. With this approach, we get control over a wider scope of functionalities, compared to the BAdI implementation. We will now try to add a similar custom button using this method.

First we need to find the name of the standard SAP Application Interface Framework class, which we will later inherit and reimplement. Execute Transaction /AIF/CUST and go to the **SAP Application Interface Framework • Error Handling • Define Applications** node. On the next screen, in the **Maintain Application** node, you should see one entry for **Application ID AIF**, as shown in Figure 78.

Note the name of the **Action Handler** class: `/AIF/CL_AIF_ACTION_HANDLER`. This class is responsible for handling most of the events occurring in the monitoring and error handling transaction. Because it is part of the customizing, we can use a different, custom class instead of the standard one. The only requirement is that our class needs to inherit from the `/AIF/CL_ROOT_ACTION_HANDLER` superclass.

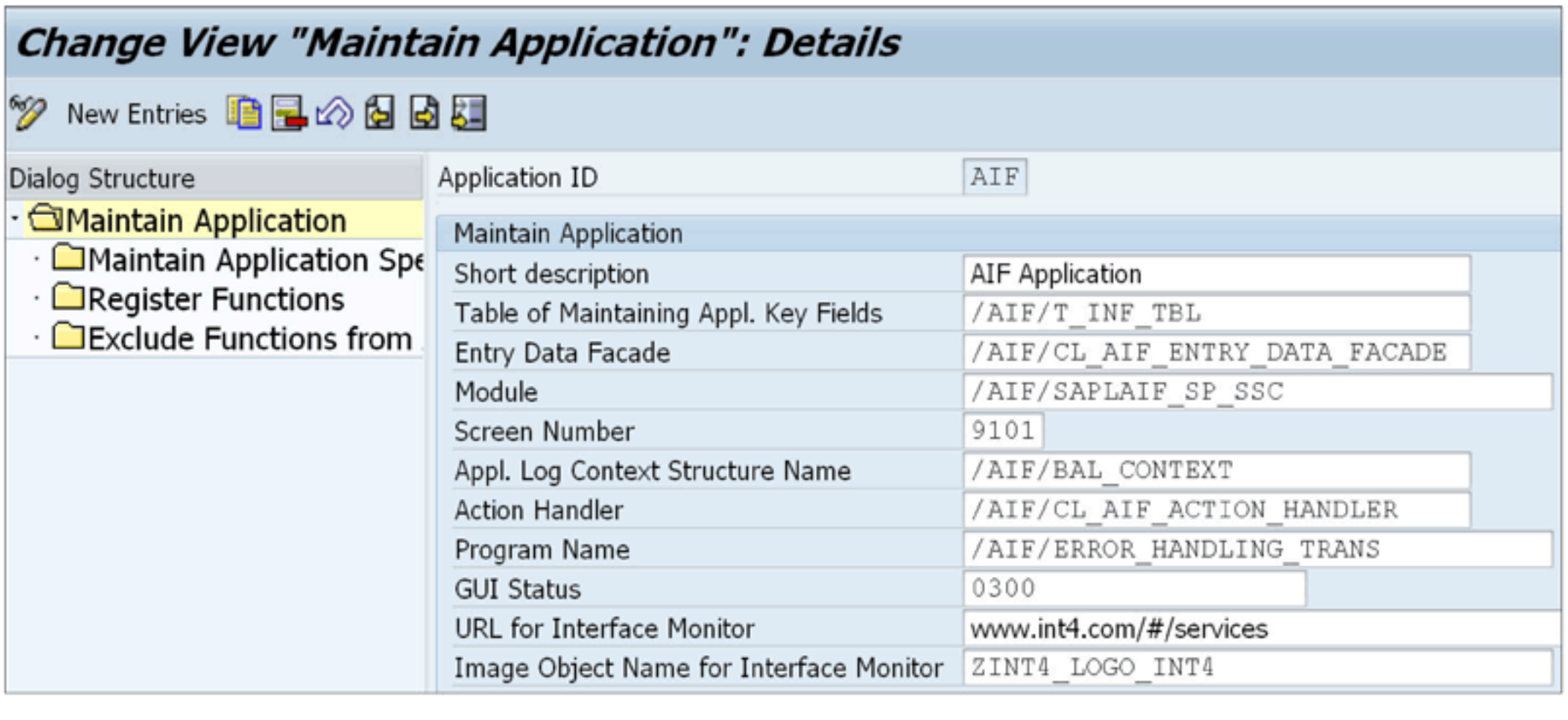

Figure 78 SAP Application Interface Framework Application ID with Standard Action Handler Class

We will now create the custom class. Go to Transaction SE24, specify the /AIF/CL_AIF_ACTION_HANDLER class, and click **Copy** or press Ctrl+F5. In the next pop-up, specify the name of your custom class as ZAIF_CL_AIF_ACTION_HANDLER_CUS and confirm. Once the class is copied successfully, specify its name in the **Object Type** field and click on **Change** to enter change mode. You now have your custom action handler class and can start making changes to all the functionalities that the class provides.

Note

The /AIF/CL_AIF_ACTION_HANDLER class is set as **Final**. That means we cannot inherit from it and need to create a copy, which means that we detach our custom solution from the standard SAP Application Interface Framework implementation of the action handler. The disadvantage is that we cannot then benefit from future changes provided by SAP as part of upgrades, support packages, or even single SAP Notes related to the standard action handler class. Consider this when choosing between the BAdI and custom class approaches.

We will begin with adding a custom button to the same *View1* window, as we did with the BAdI approach. Select the ENHANCE_GET_FUNC_LIST method and

click the **Sourcecode** button to enter the source code editor. Locate the code in Listing 3, at the beginning of the method.

```
*  remove all registered function that have a registererd
*  function class. Such buttons should only be added
*  with the corresponding method GET_FUNC_LIST->
*  Behavior should be the same as for a BADI implementation
DELETE ct_func_list WHERE reg_func_class IS NOT INITIAL.
```

Listing 3 Source Code in ENHANCE_GET_FUNC_LIST Method

On the next line, following the code from Listing 3, add the code snippet from Listing 4 at the end.

```
*{Custom button
  DATA: ls_func type /aif/t_func_lst_st.
  IF iv_vcode = 1.
    ls_func-application_id = 'AIF'.
    ls_func-func_code      = 'CUSTBUTTON2'.
    ls_func-view_id        = '1'.
    ls_func-seqno          = '001'.
    ls_func-btn_icon       = '@3Q@'.
    ls_func-description    = 'Custom button 2'.
    ls_func-tooltip        = 'Custom Button 2'.
    ls_func-disabled       = ' '.
    INSERT ls_func INTO TABLE ct_func_list.
  ENDIF.
*}Custom button
```

Listing 4 Adding Custom Button in Custom Action Handler Class

Save and **Activate** your changes. The next step is to program the logic that will execute when the button is clicked. For that, we need a new method in our class. The method does not need any parameters. The only requirement is a proper name. The logic implemented inside of the `/AIF/CL_ROOT_ACTION_HANDLER` root class reacts to any click of any button in the monitor. Then it checks the name of the button that was clicked and tries to locate a method with that name according to the pattern `ON_[button name]_[View ID]`, where `[button name]` is the name of the button clicked—in our case,

CUSTBUTTON2; and [View ID] is the number of the view in which the button was clicked—in our case, 1.

That means that when a user clicks on our new button, the root class will search for the following method and will try to execute it: ON_CUSTBUTTON2_1. Instead of implementing a new method from scratch, we will copy an existing method for a standard **Restart** button and name it according to the naming standard described previously. Find the ON_RESTART_1 method, click it, and then click the **Copy** button. In the next step, click the **Paste** button next to it. Rename the COPY_1_OF_ON_RESTART_1 method to ON_CUSTBUTTON2_1 and press Enter. Confirm in the next pop-up that you want to rename the method. **Save** and **Activate** your changes. Now, the button should act in the same way as the standard **Restart** button in Transaction /AIF/ERR.

The last step that we need to perform before being able to test it is to replace the standard action handler class with our custom one in SAP Application Interface Framework customizing. Execute Transaction /AIF/CUST and go to the **SAP Application Interface Framework • Error Handling • Define Applications** node. Replace the **Action Handler** standard class with your own class, as shown in Figure 79.

Application ID	AIF
Maintain Application	
Short description	AIF Application
Table of Maintaining Appl. Key Fields	/AIF/T_INF_TBL
Entry Data Facade	/AIF/CL_AIF_ENTRY_DATA_FACADE
Module	/AIF/SAPLAIF_SP_SSC
Screen Number	9101
Appl. Log Context Structure Name	/AIF/BAL_CONTEXT
Action Handler	ZAIF_CL_AIF_ACTION_HANDLER_CUS
Program Name	/AIF/ERROR_HANDLING_TRANS
GUI Status	0300
URL for Interface Monitor	www.int4.com/#/services
Image Object Name for Interface Monitor	ZINT4_LOGO_INT4

Figure 79 Custom Action Handler

Save the changed entry and go to Transaction /AIF/ERR to test it. Select your interface and locate any message in an error status. Select it and click your **Custom Button 2**. A restart action, which is now implemented in our custom method, should trigger, as shown in Figure 80.

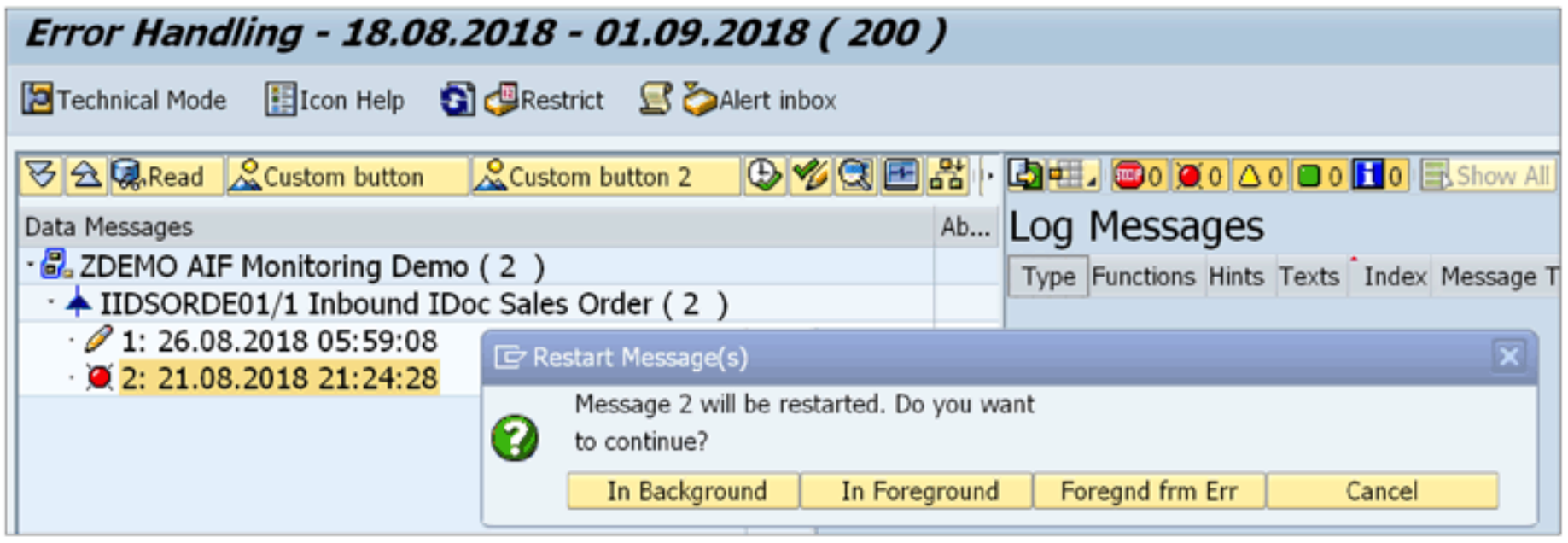

Figure 80 Restart Action Triggered by Clicking Custom Button

This concludes this section. You should now be able to further enhance your monitoring and error handling transaction using one of two methods: BAdI or custom class.

3 Interface Monitoring with Transaction /AIF/IFMON

The interface monitor is a top-level quick-access monitoring tool. Its main purpose is to provide a fast and up-to-date summary of the interfaces that are assigned to the user. Apart from the status overview, it facilitates alerts and email notifications management and enables easy navigation to more detailed information in the error handling or message summary reports.

The monitor can be accessed using Transaction /AIF/IFMON or from SAP Easy Access, following the path **Cross-Application Components • SAP Application Interface Framework • Interface Monitor**. The initial screen looks as in Figure 81.

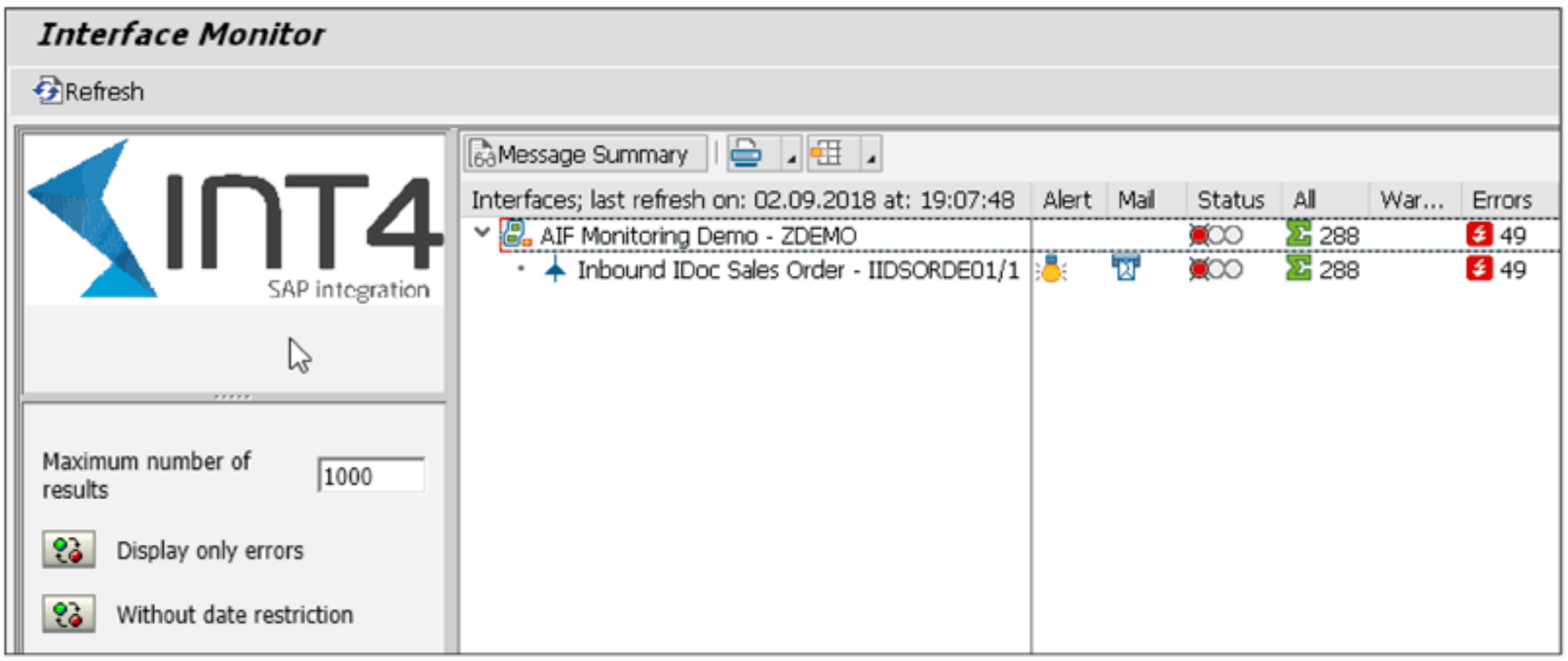

Figure 81 Interface Monitor: Screen Overview

> **Note**
>
> To display the messages of an interface, the user has to be assigned to a recipient configured for the interface (with the **Include in Overview Screen** option selected). The topic of recipients' assignment is covered in more detail in Section 4.

The interface overview is displayed in the right panel of the interface monitor (Figure 82).

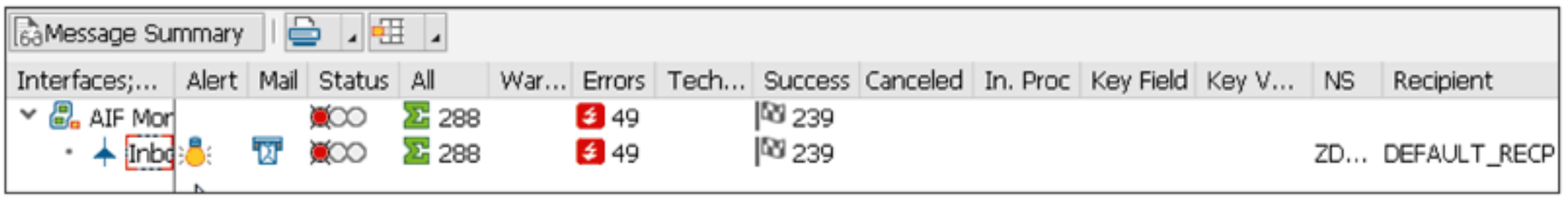

Figure 82 Interface Monitor: Overview List of Interfaces

The following is a short description of each column presented in Figure 82 (from left to right):

- **Alert**
  Lightbulb icon indicating whether new alerts exist (yellow color). If there are no new alerts or all alerts are confirmed, the lightbulb is greyed out.

- **Mail**
  Email notification icon that lets you change your own mail preferences. Three modes are available:
  - No email notifications.
  - Single email notification when new alert is created. Until the alert is confirmed, no new emails are triggered (regardless of the number of errors).
  - Email notification for every error.
- **Status**
  The standard "traffic light" icon is used to indicate interface status:
  - Green: all messages processed successfully.
  - Yellow: at least one message was processed with warning.
  - Red: at least one message was processed with an error.
  - Grey: no messages selected.
- **All**
  The number of all the messages in the selected range.
- **Warnings**
  The number of messages with warnings in the selected range.
- **Errors**
  The number of messages with errors in the selected range.
- **Technical Errors**
  The number of technical errors in the selected range. Displayed only for technical users.
- **Success**
  The number of messages processed successfully in the selected range.
- **Cancelled**
  The number of cancelled messages in the selected range.
- **In. Proc**
  The number of messages still being processed in the selected range. Displayed only for technical users.

- **Key Field**
  The name of the key field (if any) that led to the assignment to the current recipient. Empty if no key field rule is used.
- **Key Values**
  The value of the key field that led to assignment to the current recipient. Empty if no key field rule was used.
- **NS**
  The namespace where the current recipient is defined.
- **Recipient**
  The name of the current recipient.

Apart from icons already mentioned in the **Alert** and **Mail** columns, the message counts in the **All**, **Warnings**, **Errors**, **Technical Errors**, **Success**, **Cancelled**, and **In Proc** columns are also interactive. Clicking any of the numbers in the mentioned columns opens the error handling transaction. Messages displayed in error handling will be limited to the type matching the selected column and to the filter criteria from the left panel of the interface monitor (described in more detail ahead).

The left-side panel (Figure 83) accommodates elements that provide additional filtering properties. Their purpose is mainly to facilitate preconfigured switching to error handling.

- **Maximum Number of Results**
  Limits the number of messages that will be displayed when switching to error handling.
- **Display Only Errors/Display All Messages**
  Toggles whether only errors or all messages should be displayed when switching to error handling.
- **With/Without Date Restriction**
  Toggles **Calendar** display and **Date Range** screen input.
- **Date Range**
  Allows a user to specify a date range for message overview. The provided date range is also applied when switching to error handling.

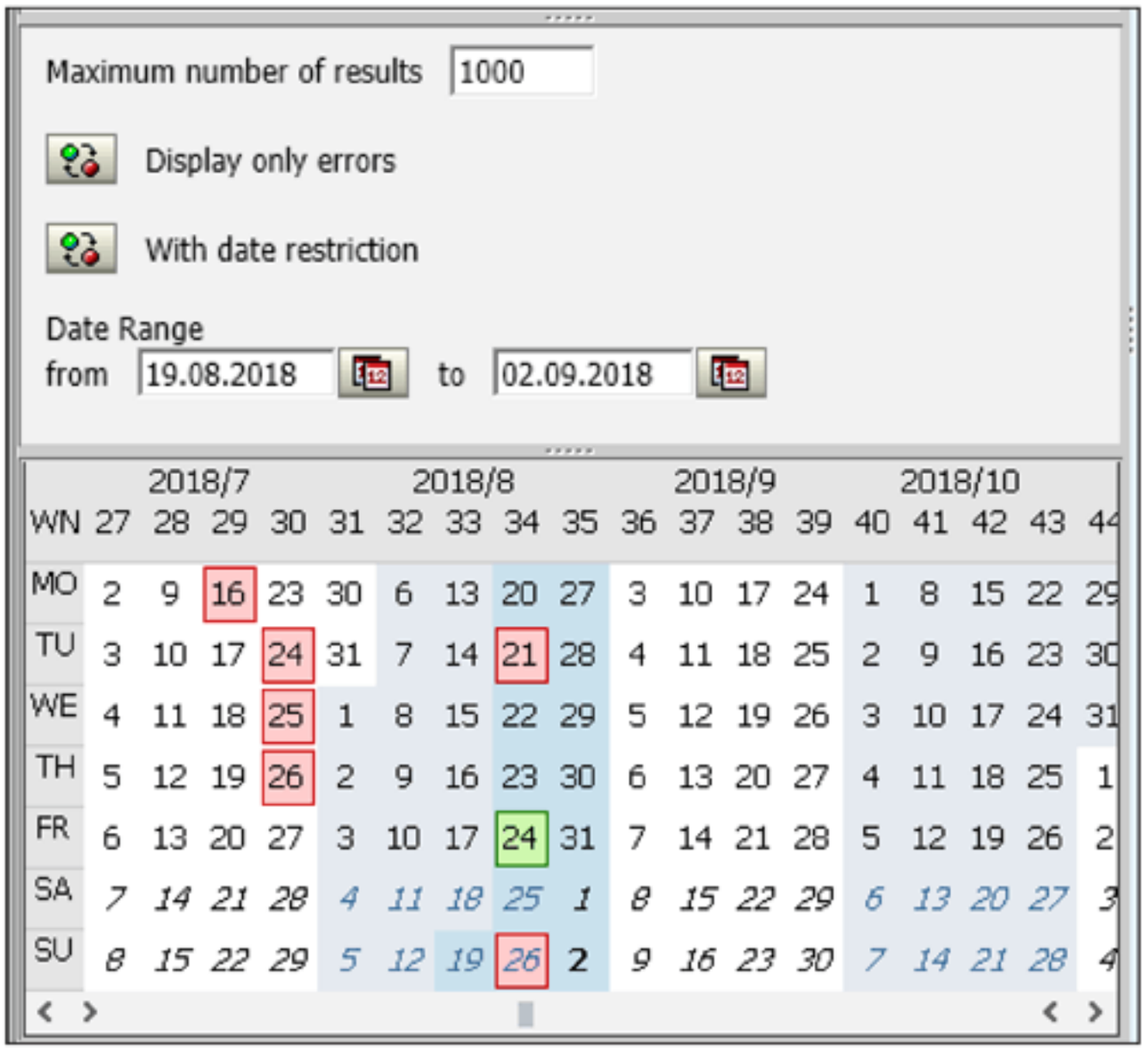

Figure 83 Interface Monitor, Right Panel: Filters and Calendar

When toggled on, the **Calendar** screen element provides an overview of the status of the interface across time. The background color of each day indicates the status of an interface. Available indicators are as follows:

- **Green**
  All messages successfully processed on given day
- **Red**
  At least one messages is in error on given day
- **No background**
  No messages were received on given day

The **Calendar** screen element is interactive. A user can select a day from a calendar, and the selected date will automatically populate the **Date Range** input. A range of dates can be selected by clicking a start or an end day and dragging the mouse cursor to a different day.

4 Alerts Management

By *alert* in SAP Application Interface Framework, we mean a notification informing its recipients that a critical situation has occurred in one or more interfaces, requiring immediate action to resolve the situation. SAP Application Interface Framework integrates strongly with SAP's alert concept, introduced by the Alert Management framework. The aim is to deliver a responsive and free of delay environment for error handling. Although it is possible to use simple SAP Application Interface Framework monitoring without any alert configuration at all, it is a prerequisite for a full-featured interface implementation. For example, to view messages in the monitor (/AIF/IFMON), a user has to be assigned to alert recipients.

> **Note**
>
> A user has to be assigned to a recipient for alerts to monitor messages, even if no alerts are being generated, to view messages that were processed correctly as well.

Alert configuration is recipient-based. A *recipient* is an abstract entity to which we can assign users or roles to include them in the monitoring process. We use recipients to define which users are responsible for which data messages. It's enough to have a single recipient defined for all the interfaces, but to accommodate complex scenarios we can have many of them, each suited to a specific role.

Alert configuration consists of three main steps:

1. Create recipients
2. Assign interfaces to the recipients
3. Assign the recipients to users or user groups

Furthermore, to assign interfaces to a recipient, one of following approaches can be followed:

- **By interface**
  SAP Application Interface Framework allows us to set up a single default

recipient for each interface. As a result, if the system can't find any other recipient for a notification, it is then sent to the default recipient. For detailed description, see Section 4.1.

- **By message categories**
  We can define message categories in SAP Application Interface Framework and specify what log messages fall under which category. Then we can use an assignment table to assign a recipient to a given message category. This allows us to group alerts into areas of responsibility and delegate them to appropriate users. For a detailed description, see Section 4.2.

4.1 Simple Alert with Default/Standard Recipient

The basic configuration is based on a concept of a default recipient who will receive all notifications for which no other recipient is found. It's sufficient for most simple scenarios. It can be set up in SAP Application Interface Framework customizing in two ways: at the namespace level as a default recipient, or at the interface level as a recipient without key fields.

Caution

Other than the default recipients, you cannot combine interface-specific recipients (recipients without key fields) with message category or key field recipients.

Configuration Steps

As a first step, we have to define recipients we want to use for the alerts. In SAP Application Interface Framework customizing (/AIF/CUST), go into **Error Handling • Define Namespace-Specific Features • Define Recipients** settings. Here we have to provide a name and optionally a description. As shown in Figure 84 for our sample configuration, we have entered "DEFAULT_RECP" as the recipient name to indicate it will serve as a default recipient.

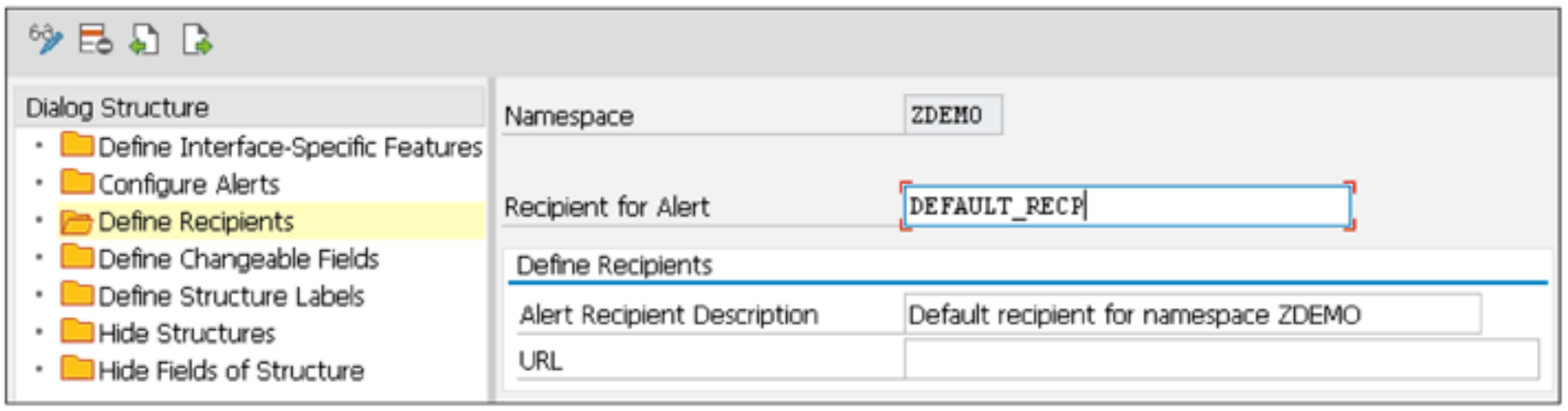

Figure 84 Define Recipients

Now we go to **Error Handling Define Namespace-Specific Features • Configure Alerts • Default Recipient** settings to set up an alert. As in Figure 85 we enter the name and version of the interface the alert will be related to. Then we also enter the namespace and name of the recipient that is to be used as the default (only one default recipient can be specified as default for given interfaces). In our case, we provide the name and namespace we used in the previous step. The rest of the fields we leave empty for now as they take part in the advanced configuration that will be covered in Section 4.2. The checkboxes for message types are related to Transaction /AIF/IFMON (the monitor) and their specific use is covered in Section 3.

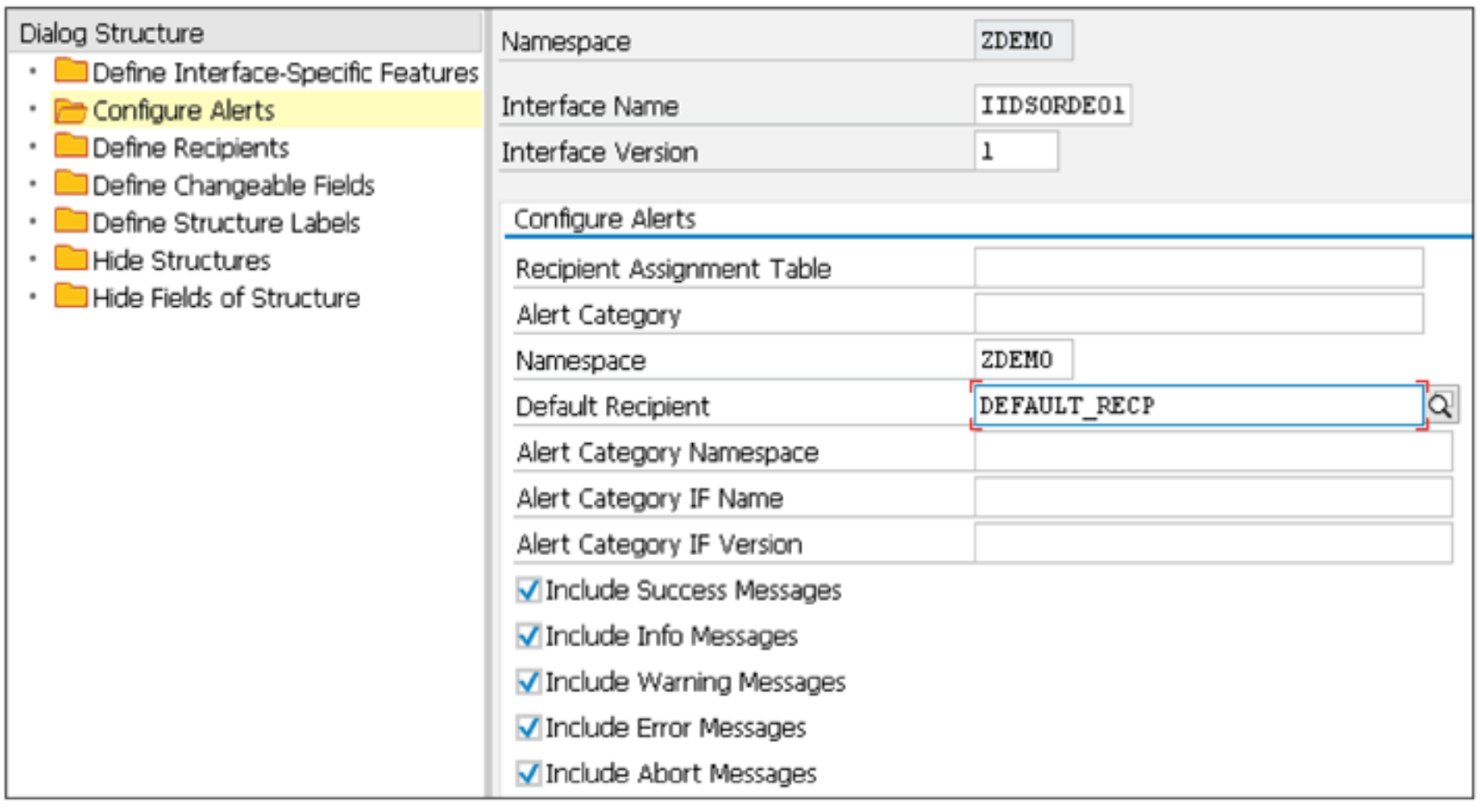

Figure 85 Configure Alerts

Alternatively (or in addition) to the default recipient set up in alert configuration at the namespace level, we can specify a recipient at the interface level with the same end result. This is the so-called recipient without key fields. To set up this recipient, go into **Error Handling • Define Interface-Specific Features • Assign Recipients without Key Fields** (Figure 86) and we enter another recipient defined as in Figure 87.

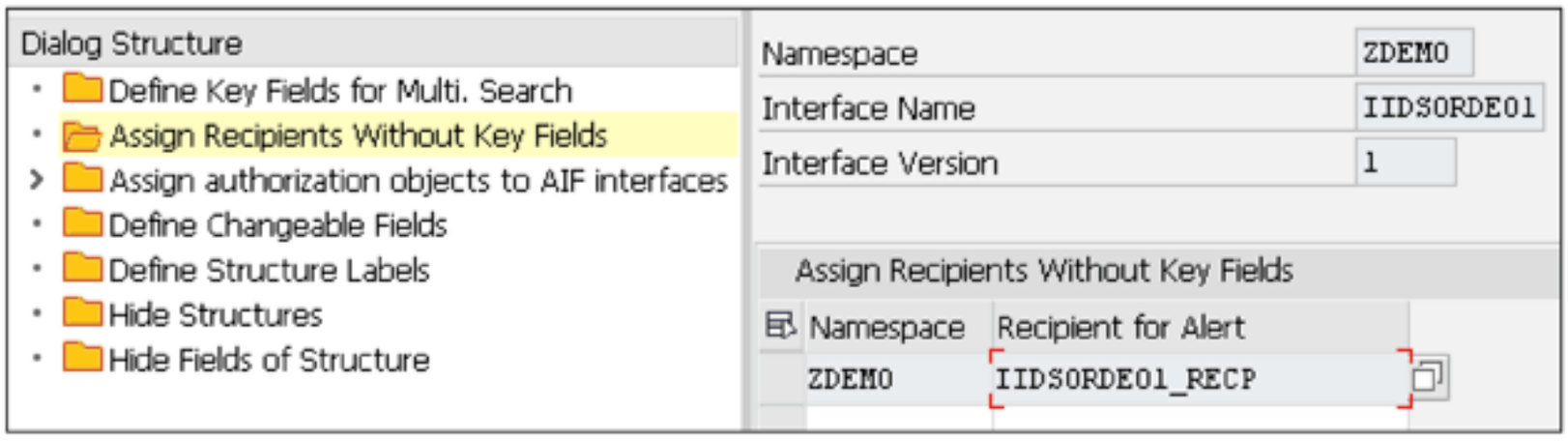

Figure 86 Assign Recipients without Key Fields

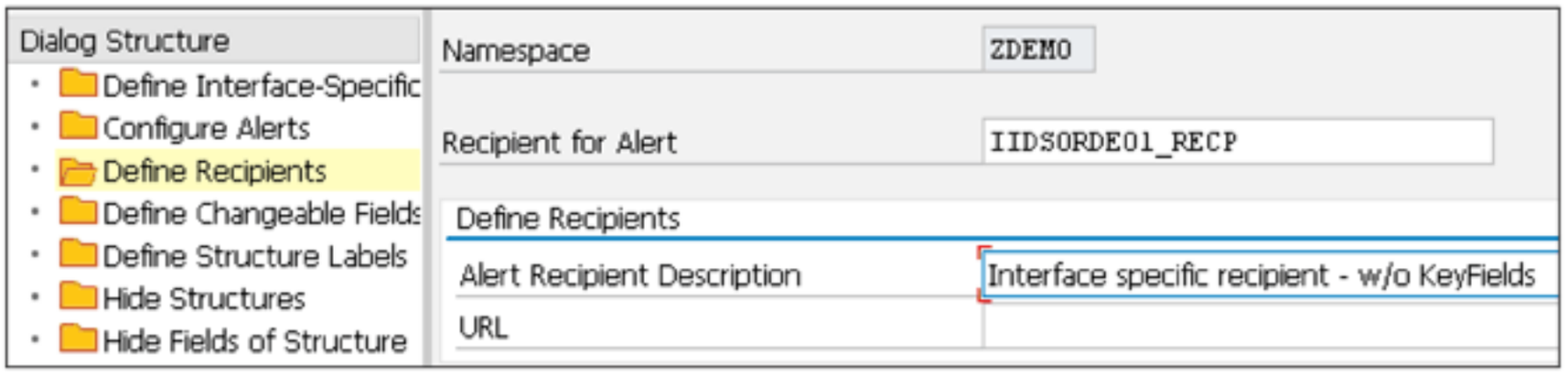

Figure 87 Define Recipients

The difference between a default recipient (assigned in namespace customizing) and a recipient without key fields (assigned in interface-specific customizing) is that the latter is ignored if an assignment table is used in **Error Handling • Define Namespace-Specific Features • Configure Alerts • Recipient Assignment Table**. This advanced setup will be described in more detail later on in Section 4.2.

When we have a default recipient or recipient without key fields (or both) assigned to an interface, we need to assign users or roles to the recipient. We do it in SAP Application Interface Framework customizing in **Error Handling • System Configuration • Assign Recipients**. In our example, we will use the **Assign Users** option as shown in Figure 88. Enter the user number and name.

From the **Message Type** dropdown, select **Application Error or Technical Error**. Let's also mark the **Include on Overview Screen** checkbox (to see messages in the monitor) and the **Technical User** checkbox.

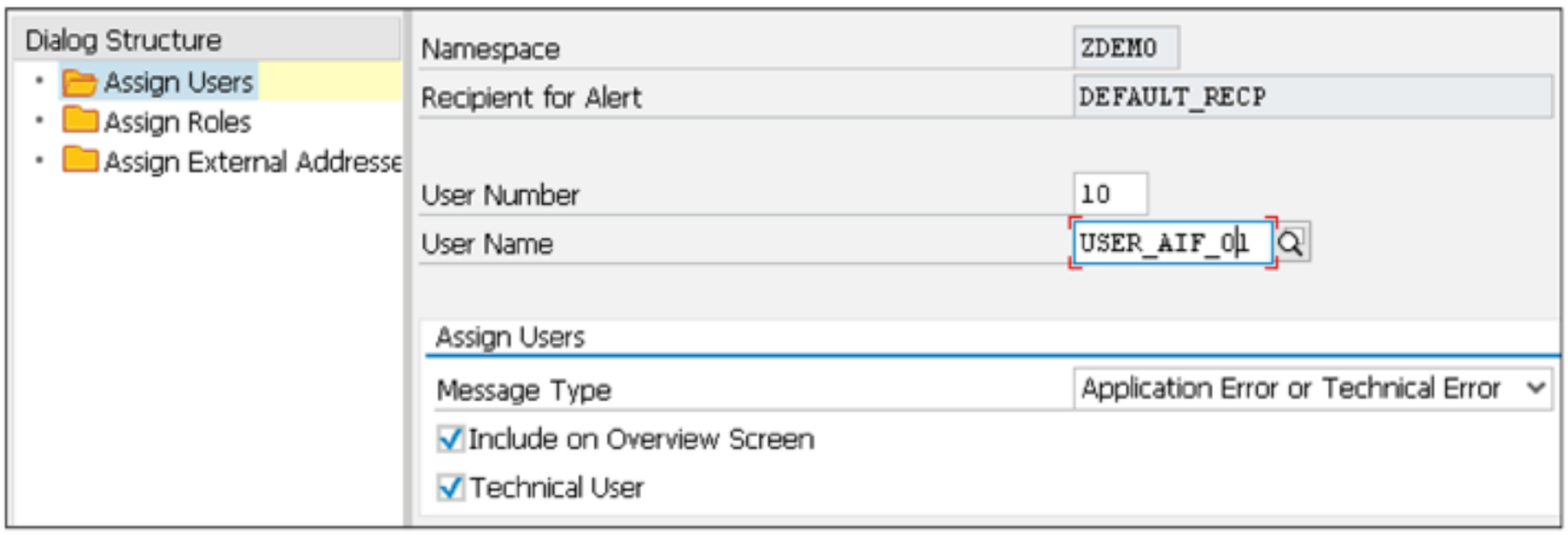

Figure 88 Assign Users

This customizing is not transportable and can be maintained on each system to allow for flexibility with assigning test and productive users.

Test

To test the alert configuration for the default recipient setup, let's send an ORDERS05 IDoc using Transaction WE19 as shown in Figure 89. Because we want to check SAP Application Interface Framework's reaction to an error situation, we will enter a nonexistent material number.

We check the processing results in the monitor (as seen in Figure 90). As expected, the IDoc failed. In the **Log Messages** window, in addition to actual error information (incorrect material), we can also see information that an alert was created. In this case, the alert was assigned ID 00009 and belongs to default category /AIF/ALERT_CAT_DEF. Note that messages of type **Information** have to be enabled in the log messages view.

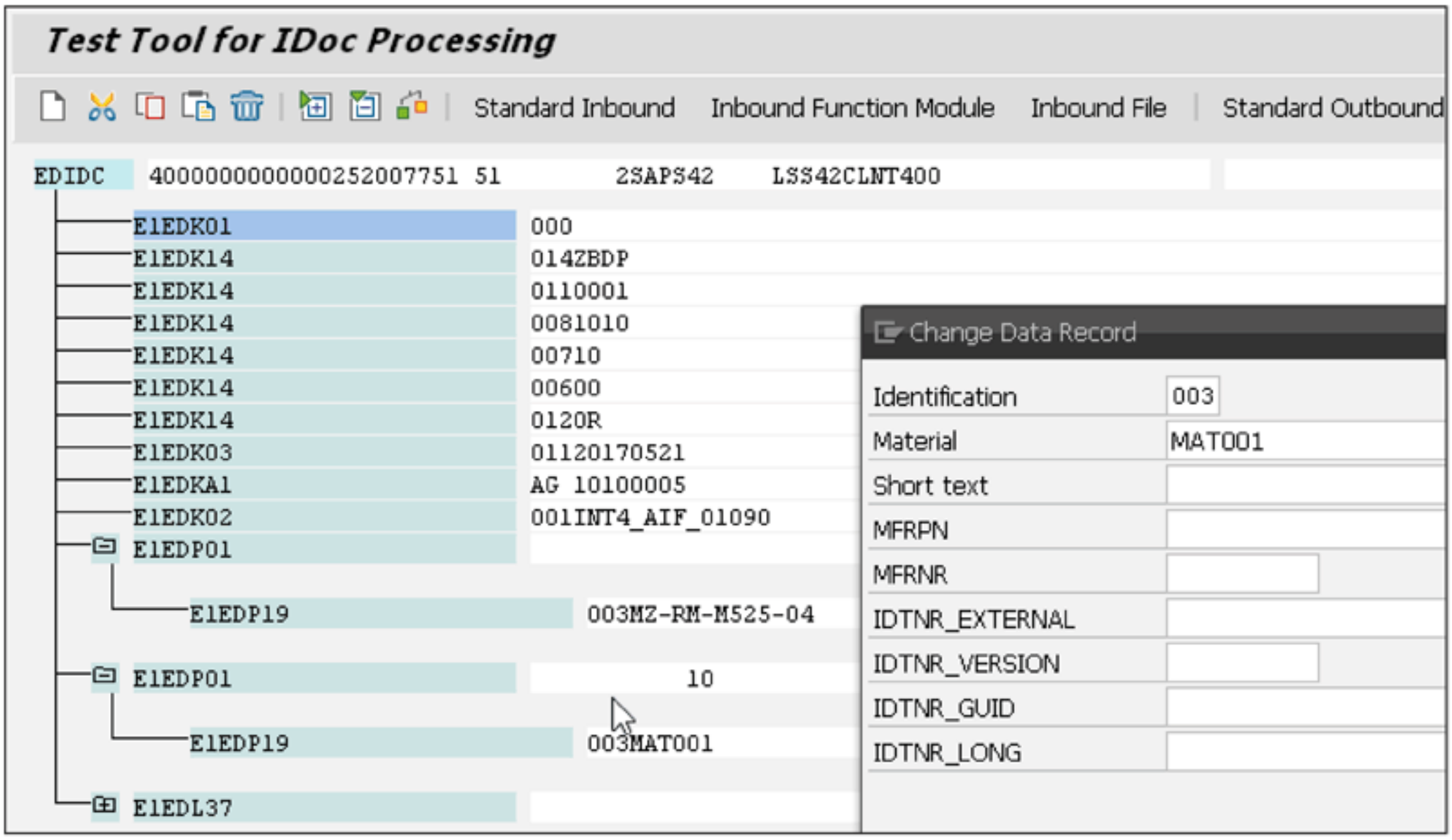

Figure 89 Test IDoc ORDERS/ORDERS05

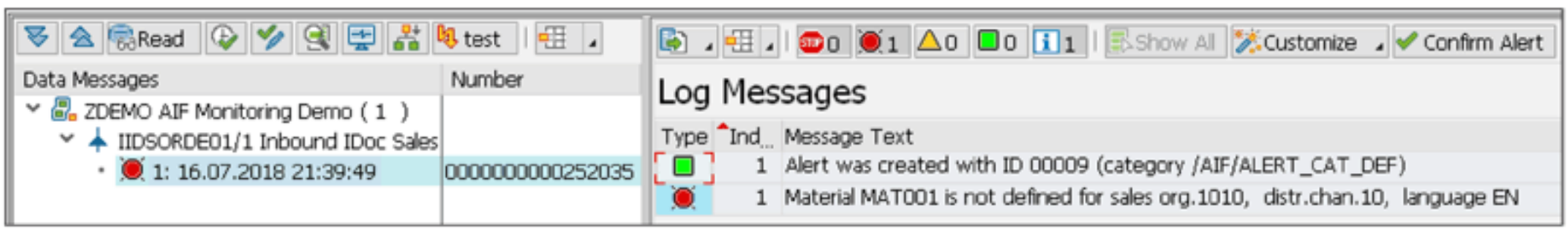

Figure 90 Interface and Error Hanlding Monitor (Transaction /AIF/ERR)

As a result of setting up a default recipient and assigning to it our user, we enabled display of messages in the monitor. Also, we can see now, as in Figure 91, a lightbulb icon in the **Alert** column, indicating a new alert was created for this interface. You can learn more details about the monitor in Section 3.

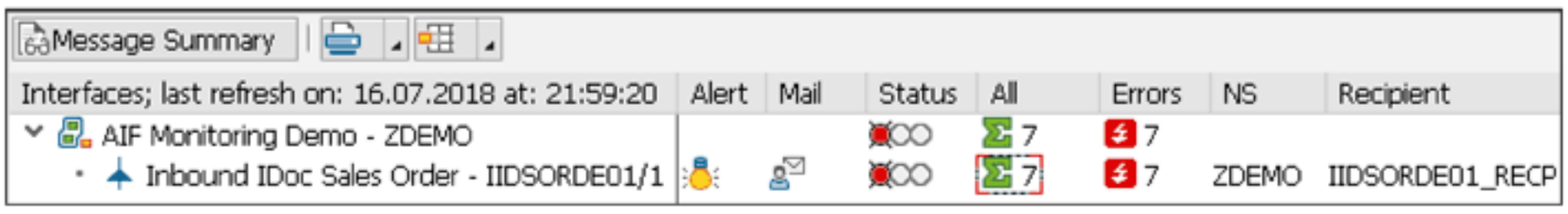

Figure 91 Interface Monitor (Transaction /AIF/IFMON)

Furthermore, as our user's profile includes an email address, we will also see an email prepared for distribution in Transaction SOST (Figure 92).

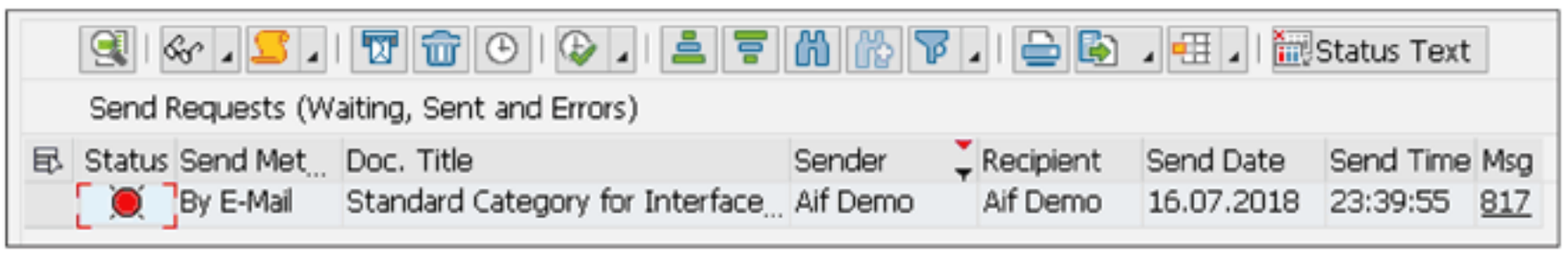

Figure 92 Send Request Transaction (SOST)

At this point, the content of the email as shown in Figure 93 is very limited as it's based on the default configuration for pre-delivered alert category /AIF/ALERT_CAT_DEF.

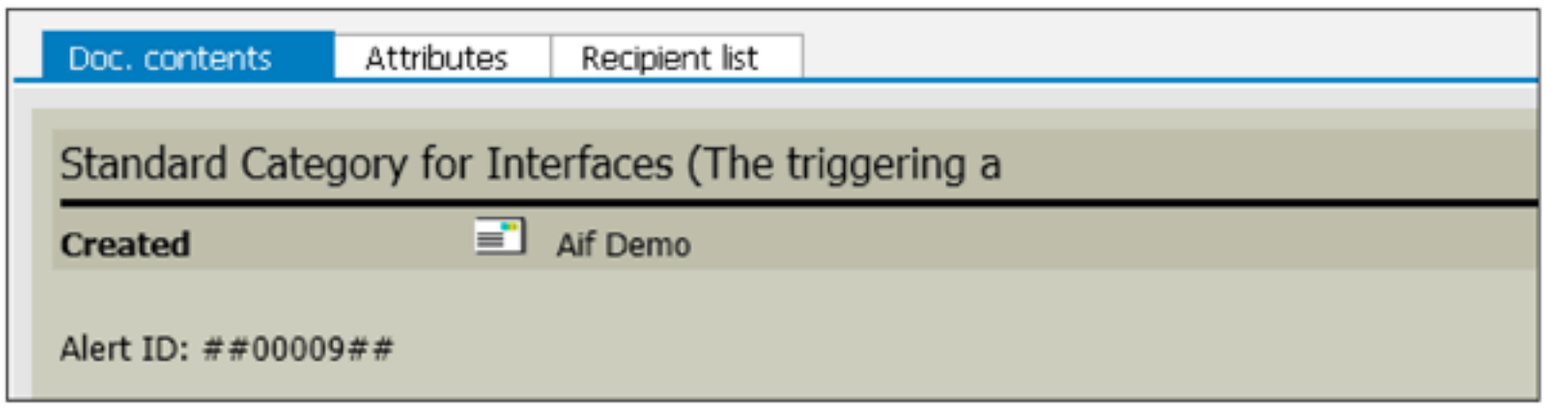

Figure 93 Email Message Content (SOST)

> **Note**
>
> The details of email content configuration for alerts (performed in Transaction ALRTCATDEF) are not covered in this E-Bite.

4.2 Advanced Rules for Alert Routing

If a single recipient per interface approach is not enough and a more sophisticated solution is required, you can look into recipient assignments driven by message category. The message categories are defined in SAP Application Interface Framework and group specified log messages together to be handled by a dedicated recipient. For example, you could have separate recipients for business errors and for technical ones. If the default recipient is also defined, it will get all messages regardless of category assignment.

What's more, you can specify a fallback recipient as well to catch only those messages that don't fall under any of the defined categories.

Assign Recipients by Message Category: Configuration Steps

First we have to create a custom table that we will use to maintain message categories and assigned recipients. For this purpose, we copy predelivered template table /AIF/RECA_TMPL in Transaction SE11 (Figure 94).

Transparent Table: ZAIF_RECP_ASGN — Active
Short Description: Copy of /AIF/RECA_TMPL

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields

Srch Help | Built-In Type | 1 / 12

Field	Key	Ini...	Data element	Data Type	Length	Deci...	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	Client
.INCLUDE	☑	☑	/AIF/ALRT_IF_KE...	STRU	0	0	AIFDefault: Interface Keyfileds
NS	☑	☑	/AIF/NS	CHAR	6	0	Namespace
IFNAME	☑	☑	/AIF/IFNAME	CHAR	10	0	Interface Name
IFVERSION	☑	☑	/AIF/IFVERSION	CHAR	5	0	Interface Version
.INCLUDE	☑	☑	/AIF/ALRT_CAT_S...	STRU	0	0	Category information structure (be append to tab.)
NSRECIP	☑	☑	/AIF/NS	CHAR	6	0	Namespace
RECIPIENT	☑	☑	/AIF/ALRT_REC	CHAR	30	0	Alert Management Recipient
.INCLUDE	☑	☑	/AIF/MSG_CATEGO...	STRU	0	0	Message Category
/AIF/NSMSGCAT	☑	☑	/AIF/NS	CHAR	6	0	Namespace
/AIF/MSGCATEGORY	☑	☑	/AIF/MSG_CATEGO...	CHAR	20	0	Message Category
/AIF/IS_FALLBACK	☐	☐	/AIF/IS_FALLBACK	CHAR	1	0	Fallback Recipient

Figure 94 Custom Recipient Assignment Table (SE11)

Under **Error Handling • Define Namespace-Specific Features • Configure Alerts**, populate the **Recipient Assignment Table** field with the name of the custom table we've just created as shown in Figure 95.

Go to Transaction /AIF/MSG_CAT_DEF to define message categories and provide the name of a category and its description. In our example, we will use MSG_BUS for errors related to business logic and MSG_TECH for those related to technical issues as in Figure 96.

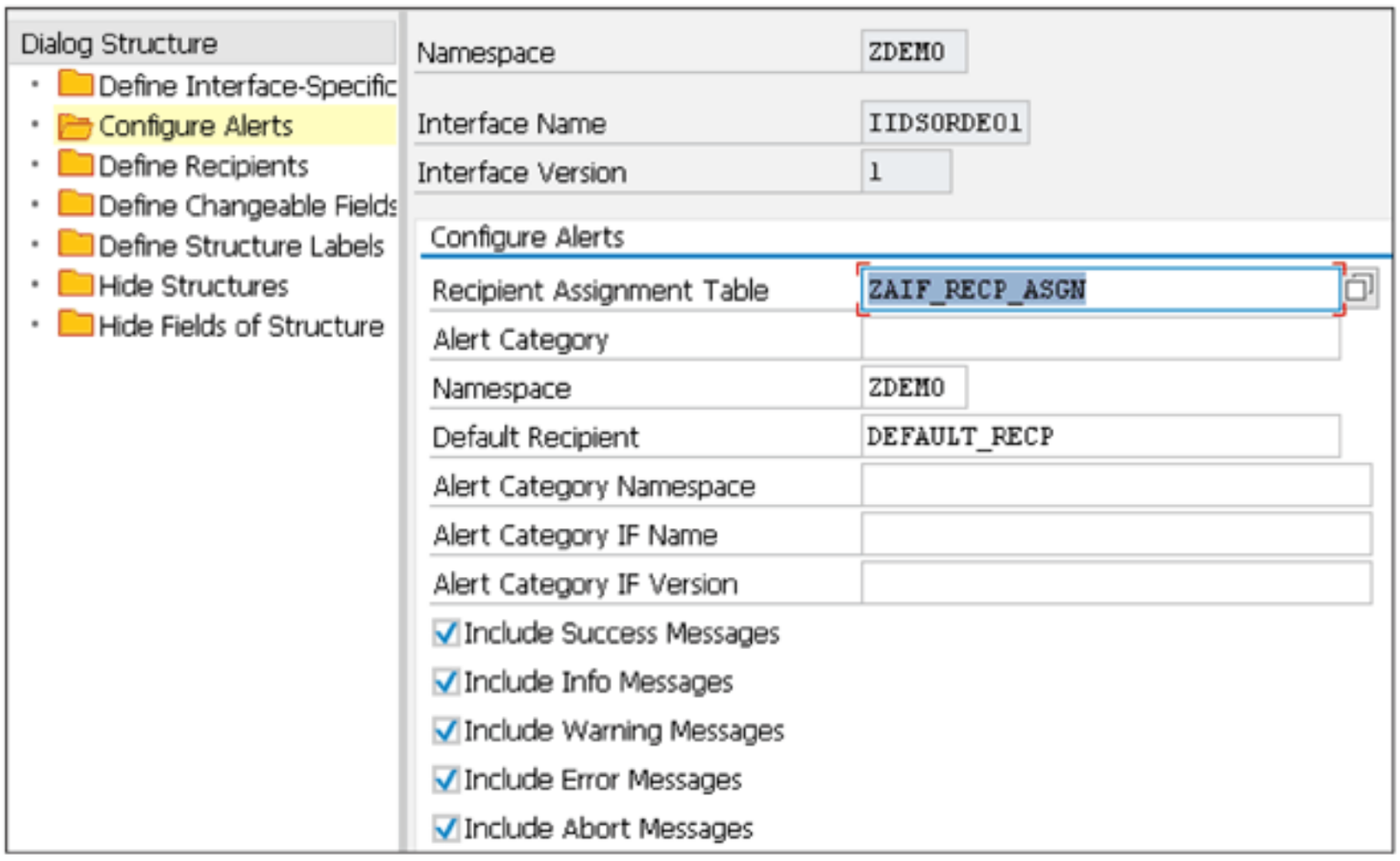

Figure 95 Configure Alerts

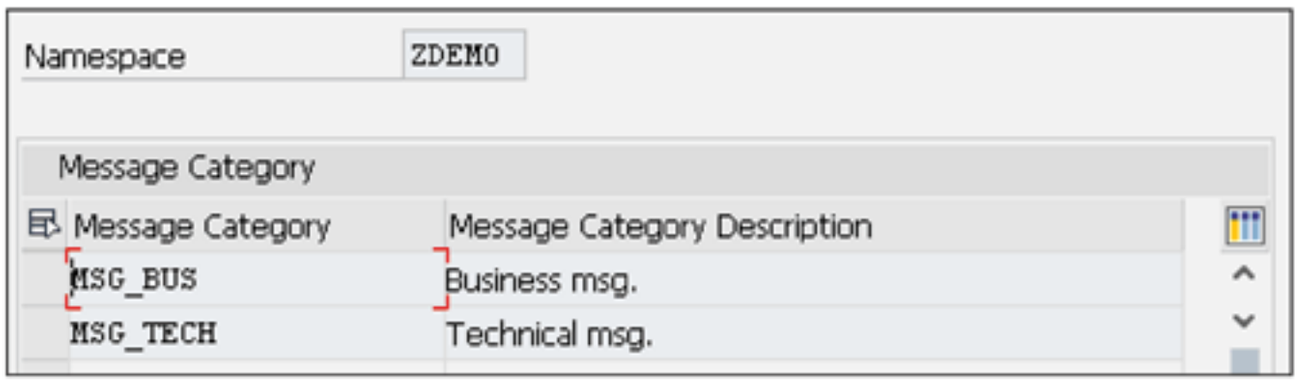

Figure 96 Define Message Categories (/AIF/MSG_CAT_DEF)

Now we have to specify precisely what SAP messages should fall under which category (from the two we defined in previous step). In Transaction /AIF/MSG_CAT_ASGN, we assign a message of class /AIF/MES, number 038 to category MSG_TECH as in Figure 97.

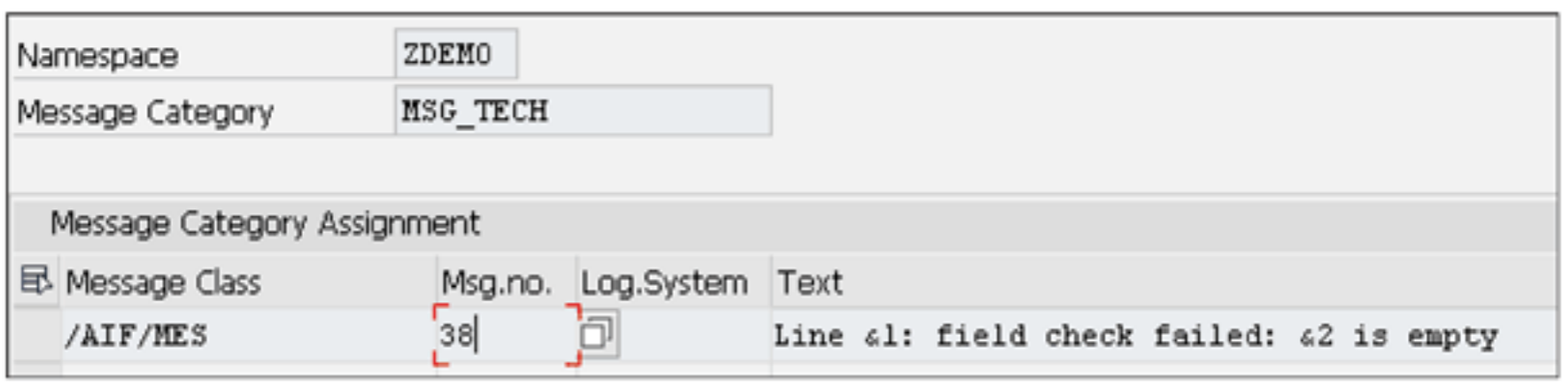

Figure 97 Assign Message Categories (/AIF/MSG_CAT_ASGN)

To category MSG_BUS, we assign message of class V1, number 381 as in Figure 98.

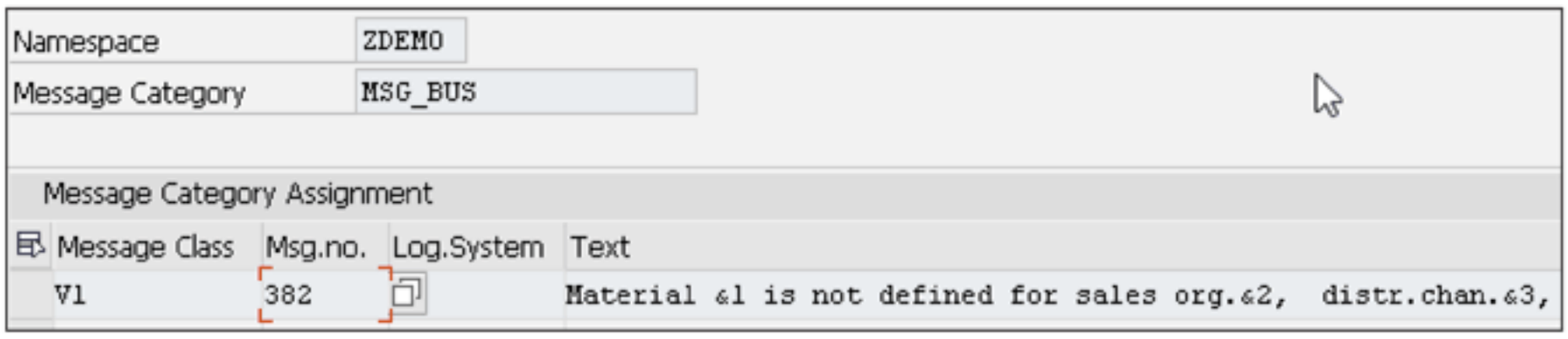

Figure 98 Assign Message Categories (/AIF/MSG_CAT_DEF)

We will create additional recipients in SAP Application Interface Framework customizing again under the **Error Handling • Define Namespace-Specific Features • Define Recipients** settings (Figure 99):

- BUS_RECP for business-related errors
- TECH_RECP for technical-related errors
- FALLBACK_RECP for errors that don't fall under the other categories

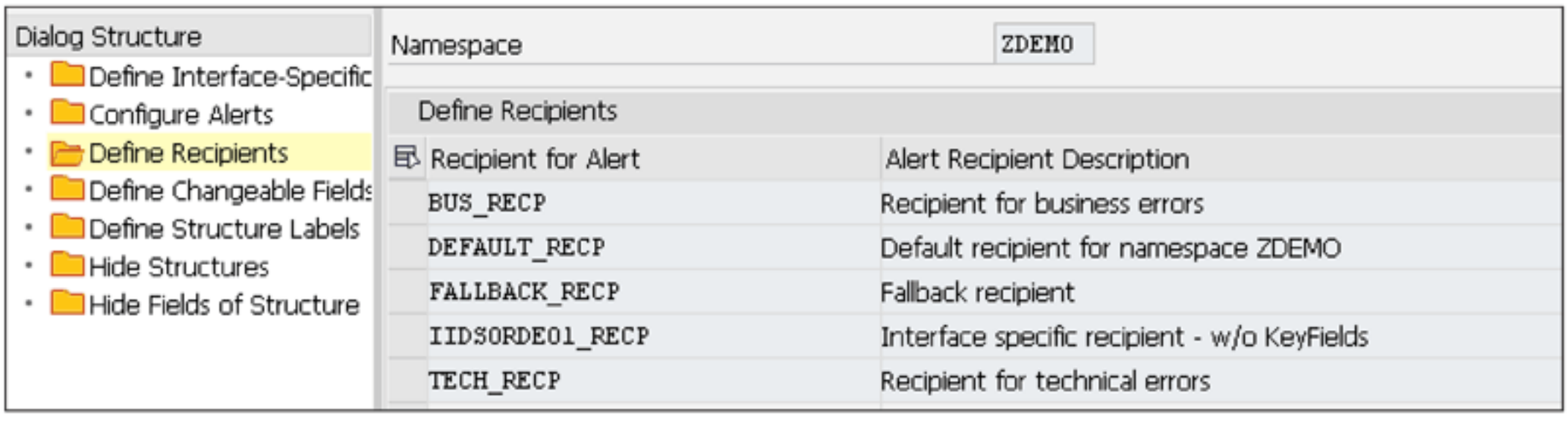

Figure 99 Define Recipients

Now we can populate the custom recipient assignment table we've created with the following values:

- Interface namespace
- Interface name
- Interface version

- Recipient namespace (for our example, it's the same as the interface namespace)
- Recipient name
- Message category namespace (for our example, it's the same as the interface namespace)
- Message category
- Fallback flag

We create three rows, two for each message category we've created and a third one for a fallback recipient for the MSG_TECH category (we could of course set up a fallback recipient for the MSG_BUS category as well). See Figure 100 for reference.

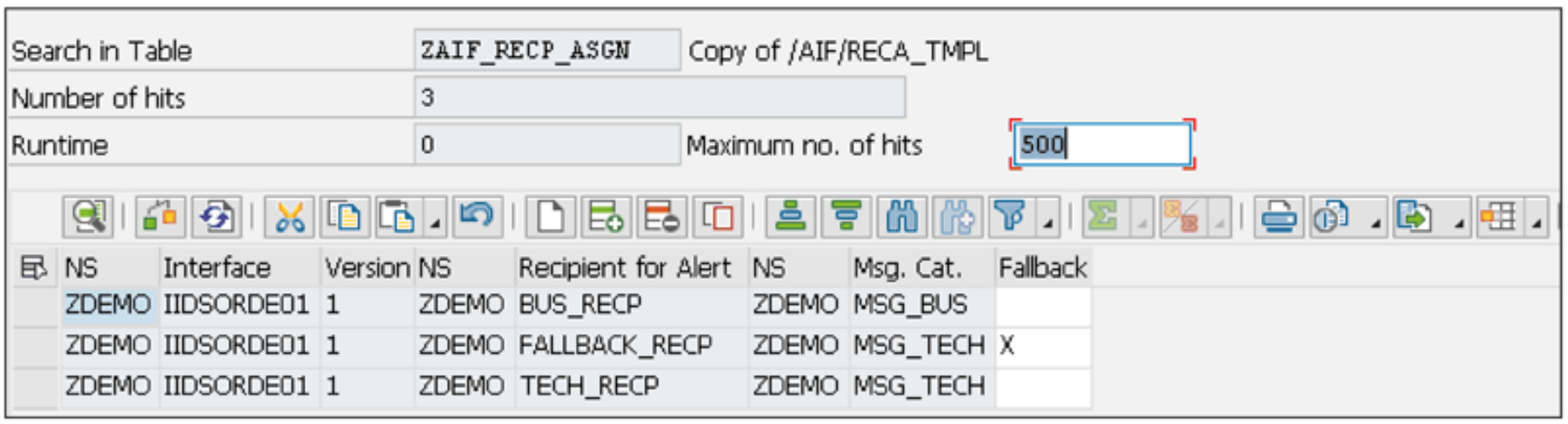

Figure 100 Populate Custom Recipient Assignment Table

Assign Recipients by Message Category: Test

To test the alert configuration for recipients assigned by alert category, let's send an ORDERS05 IDoc using Transaction WE19. We want to test two scenarios, one for each of the message categories we've defined. For the first test, let's send an IDoc with an empty document number (the **BELNR** field; see Figure 101).

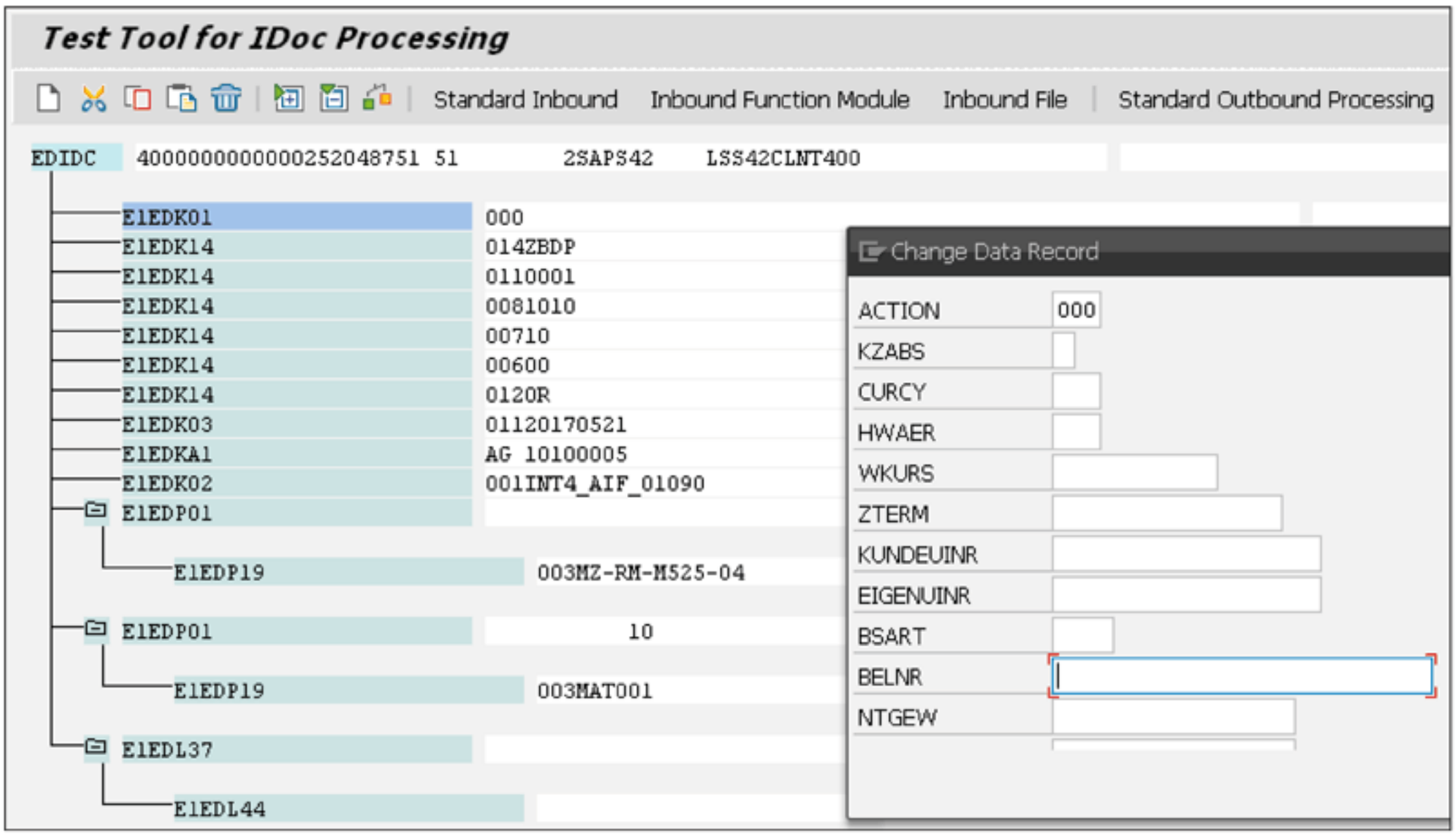

Figure 101 Test IDoc ORDERS/ORDERS05 (WE19)

The IDoc was created (with number 252050) and passed for processing (as seen in Figure 102).

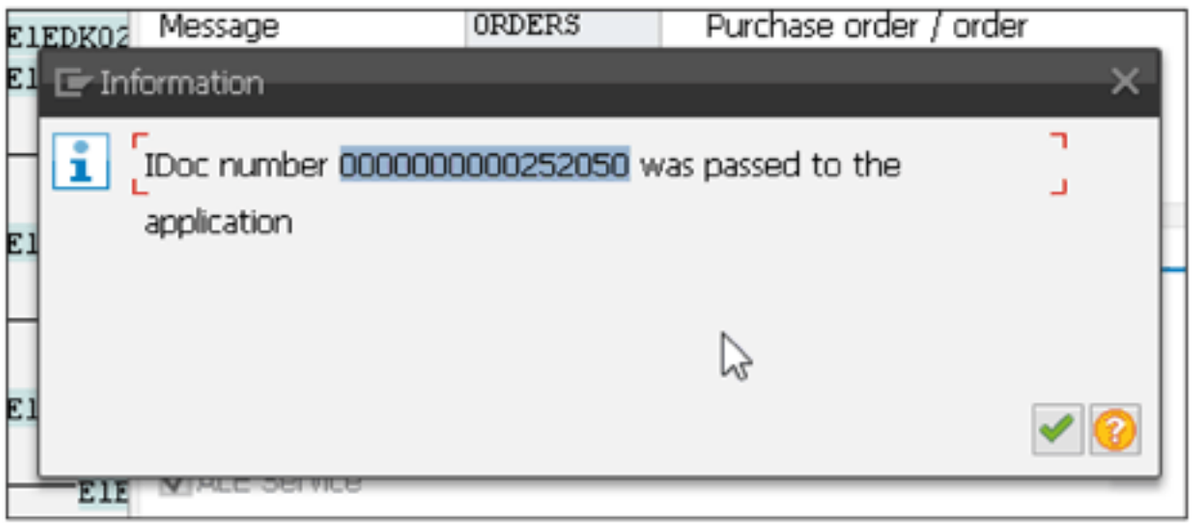

Figure 102 Test IDoc ORDERS/ORDERS05 (WE19): Confirmation

Check status of the IDoc in the monitor. As shown in Figure 103 it failed as expected, with an error message of class `/AIF/MES` and number 038 (matching one of the entries for message category assignment). We can also see an information message that an alert was created.

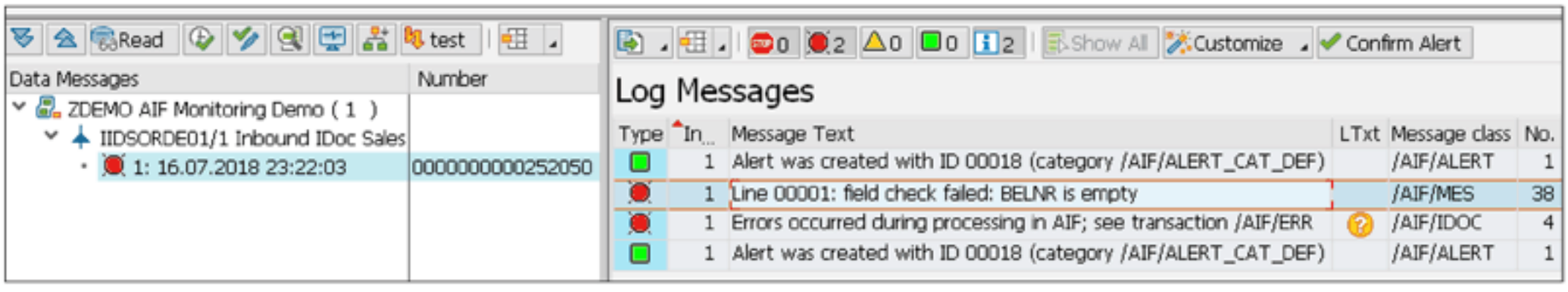

Figure 103 Interface and Error Handling Monitor (/AIF/ERR)

We want to check if the correct recipient was assigned as per the rule in our custom recipient assignment table. The easiest way is to check the entry in table /AIF/ALERT_IDX by looking at the message GUID (IDoc number in our case). As expected, the recipient is TECH_RECP, as seen in Figure 104.

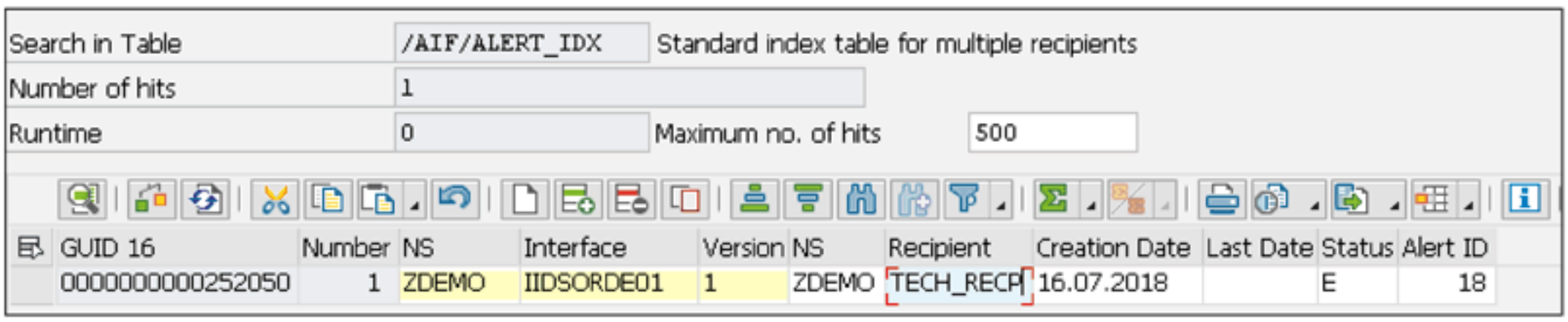

Figure 104 Starndard SAP Application Interface Framework Alert Table

To test the other scenario, send another ORDERS05 IDoc using Transaction WE19 as shown in Figure 105. This time, keep the **BELNR** field populated, but make sure that you use a material number that doesn't exist.

The IDoc again was created (with number 252052 as shown in Figure 106) and passed for further processing.

For this case, as shown in Figure 107, the error message in the monitor is of class V1 and with number 382 (matching one of the entries for message category assignment). We can also see information message that an alert was created.

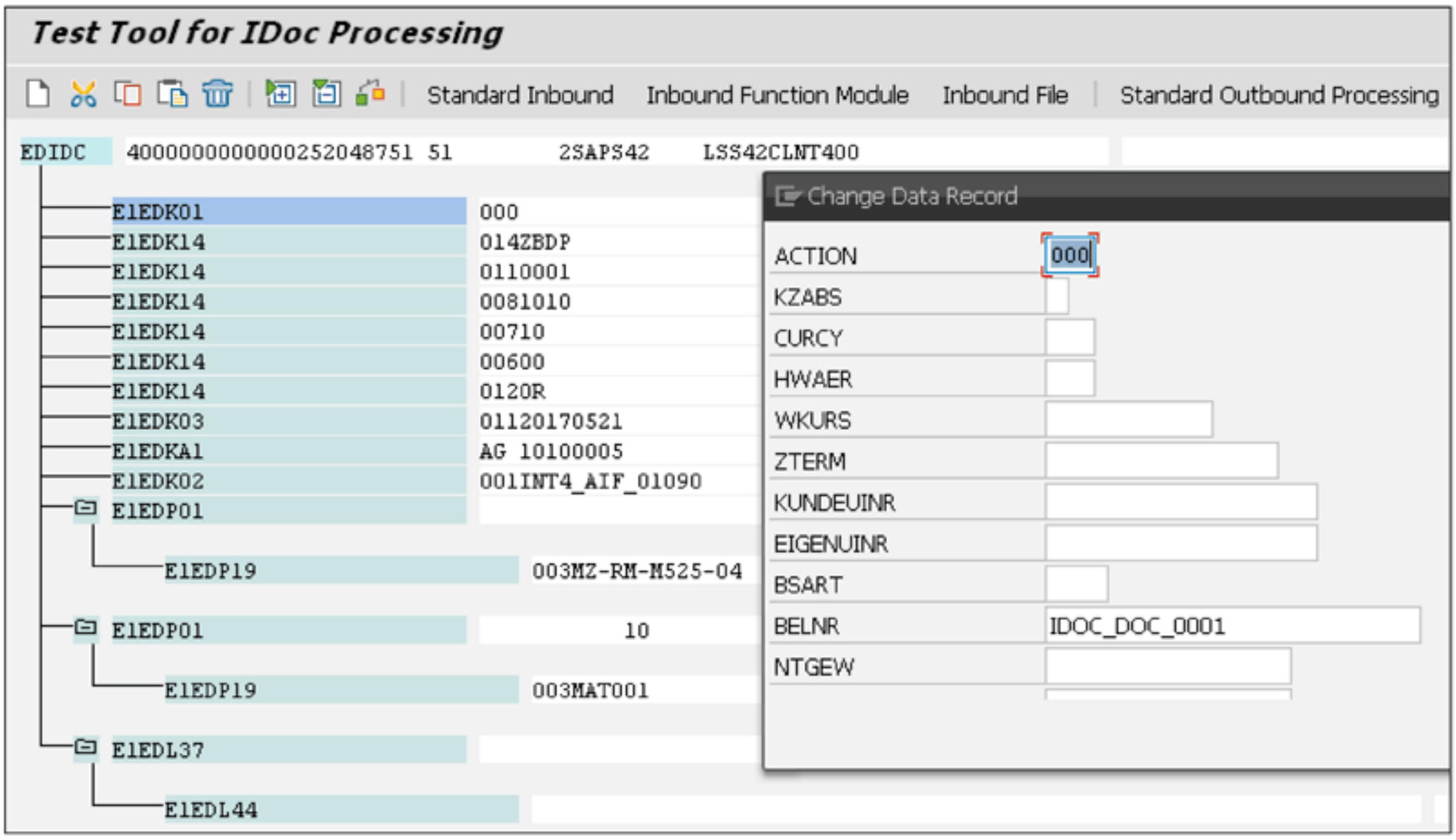

Figure 105 Test IDoc ORDERS/ORDERS05 (WE19)

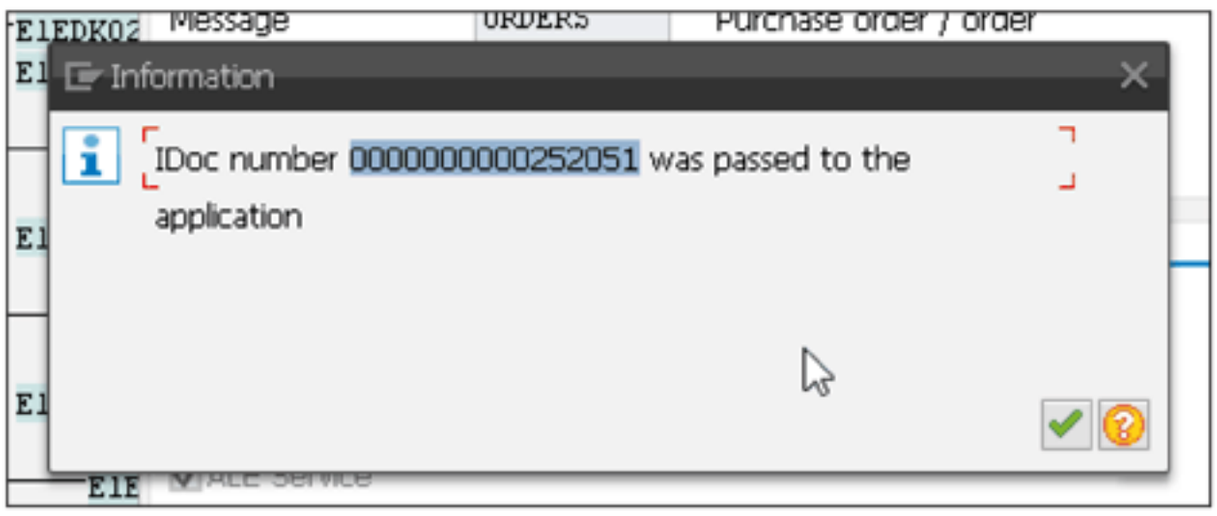

Figure 106 Test IDoc ORDERS/ORDERS05 (WE19): Confirmation

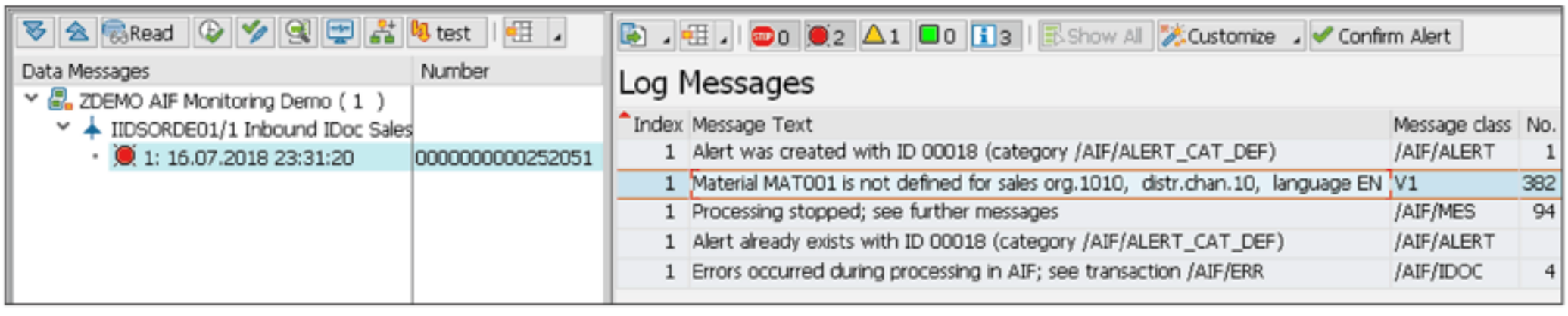

Figure 107 Interface and Error Handling Monitor (/AIF/ERR)

Again, check table /AIF/ALERT_IDX. Using the message GUID (IDoc number), we find the corresponding entry. BUS_RECP is correctly displayed as the recipient for the alert (Figure 108).

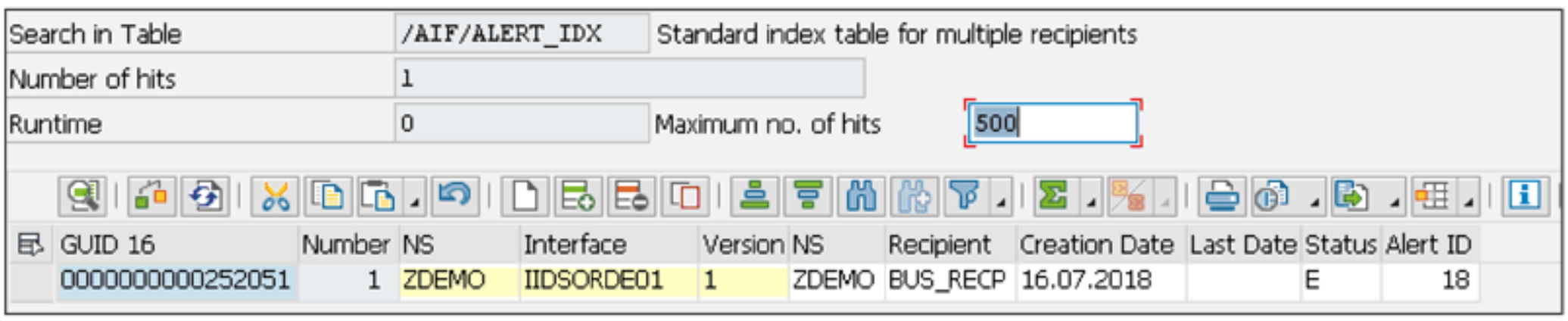

Figure 108 Standard Alert Table

5 Statistical Reports

The monitoring and error handling and the interface monitor transactions give you a good overview of the condition of your interfaces. SAP Application Interface Framework also provides additional reports that can help you further analyze the situation. In this section, we will describe two reporting solutions provided with SAP Application Interface Framework: message summary and snapshots.

5.1 Message Summary Report

After reading Section 3, you should be familiar with the interface monitor of SAP Application Interface Framework. From this transaction, you can also access an additional report called Message Summary. The main purpose of this report is to provide you with information on the most common errors that your interfaces receive. Thanks to this tool, you can quickly identify the main "pain points" of the messages and prioritize the error handling tasks.

To access the report, go to Transaction /AIF/IFMON. In this transaction, select one of the interfaces from the list and click the **Message Summary** button, shown in Figure 109.

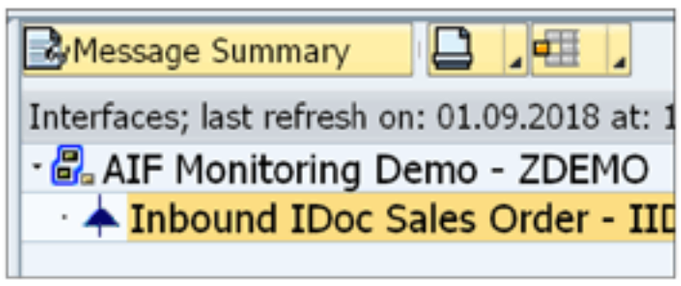

Figure 109 Accessing Message Summary Report

You will be redirected to the statistical report, which will present all the log messages that were generated for your interface messages, sorted according to the number of occurrences in the selected interface.

In the header section, you'll see the following fields:

- **Namespace/Interface/Version**
  Describing the interfaces which are the subject of the report
- **Data Messages**
  Number of data messages analyzed by the report
- **Log Messages**
  Number of log messages analyzed
- **Max.Count**
  Max number of messages to be analyzed

In the lower section, you'll see a list of sorted log messages, starting from the one that occurred most frequently.

Report columns (described from left to right):

- **Image for tree**
  Icon used when displayed in tree hierarchy
- **Application Log Messages**
  Number of application log messages for the given message class and number
- **Data Messages**
  Number of application log messages for the given message class and number

- **Message Type**
  Message type: **S-Success**, **E-Error**, **W-Warning**, **I-Info**, or **A-Abort**
- **Message Class**
  Message class of the log message
- **Message Number**
  Message number of the log message
- **Message Text**
  Text assigned to the log message
- **Logical System**
  Logical system
- **Namespace**
  Interface namespace
- **Interface**
  Interface name
- **Version**
  Interface version
- **Namespace**
  Recipient namespace
- **Recipient**
  Recipient name

Data of the report can be sorted, filtered, and exported. In addition, the following mass processing actions are provided; you can select a single line or multiple lines, and the action is carried out for all data messages of selected lines:

- **Mass Restart**
  Restart all messages matching the selected lines
- **Mass Cancel**
  Cancel all data messages matching the selected line

Looking at the report in Figure 110, you can see that the most common problem for our inbound order interface is the incorrect material number

sent in the incoming messages. This can suggest that there is something wrong with the material number mapping function (if we are using one) or that some of the customers are using incorrect material numbers when sending orders to our system.

Message Summary

Namespace:	ZDEMO	AIF Monitoring Demo
Interface:	IIDSORDE01	Inbound IDoc Sales Order
Version:	1	
Data Messages:	287	
Log Messages:	229	
Max. count:	1.000	

Mass restart | Mass cancel | Processors

Image	Appl Log	Messages	Msg type	Message ID	Msg. No.	Message text	Log.System	NS	Interface	Version	NS	Recipient
	46	46	E	V1	382	Material $1 is not defined for sales org.$2, distr.chan.$3, language $4		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	35	35	I	B1	042	Direct call started		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	21	21	W	/AIF/MES	068	Function $1 was not executed successfully		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	21	21	I	/AIF/MES	094	Processing stopped; see further messages		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	21	21	I	/AIF/ALERT	000	Alert already exists with ID $1 (category $2)		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	19	19	I	/AIF/ALERT	001	Alert was created with ID $1 (category $2)		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	17	17	E	SALERT	204	System error: $1 $2		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	16	16	I	/AIF/ALERT	003	Recipient list not found for interface $1/$2/$3		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	13	13	E	E0	332	EDI: Partner profile does not exist		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	6	6	E	VP	199	No customer master record exists for sold-to party $1		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	4	4	E	E0	414	Entry in inbound table not found		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	3	3	E	VG	204	VKORG, VTWEG, SPART cannot be determined for customer $1, vendor $2		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	3	3	E	/AIF/MES	038	Line $1: field check failed: $2 is empty		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	2	2	I	/AIF/ALERT	002	Recipient list $2 (namespace $1) cannot be determined		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	1	1	E	VP	197	Sold-to party $1 not maintained for sales area $2 $3 $4		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP
	1	1	E	VG	206	No customer number in IDoc $1 can be determined		ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP

Figure 110 Message Summary Overview

There is also a **Processor** button available that navigates to the Details and Processor Assignment report (Figure 111) for a selected line. The report provides more details on selected log messages, including specific values for log variables that the message summary ignores for statistics purposes. The totals are provided then for specific log messages with matching variable values.

In addition, for each group of messages, additional data can be provided to serve informative purposes. A processor can be assigned in the **Processor** column. Similarly, a processing status can be set from a dropdown list in the **Processing Status** column. In the **Comment** column, a free-text comment can be entered to provide more details on reasons for or solutions to the problem.

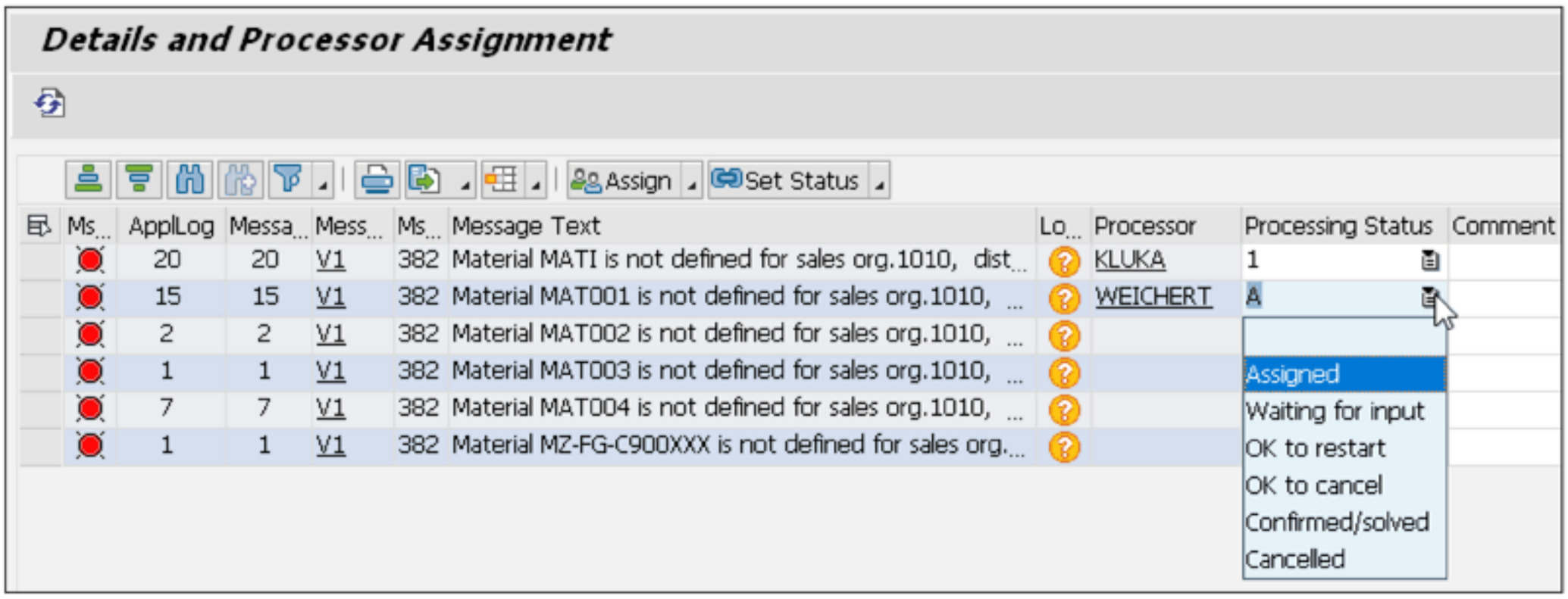

Figure 111 Details and Processor Assignment Report: Overview

5.2 Snapshots

Although the Message Summary report informs you of the current problems with your interfaces, the Snapshot report has a different goal. Using this feature, we can create snapshots of the following:

- The interface message status overview, which presents similar information to that provided in the interface monitor.
- The message summary, which presents information on the most common log messages, like the Message Summary report described in Section 5.1.

Both reports are generated for a specific day. If we do this frequently, we can then analyze a trend for our interfaces. We will see how the number of messages sent per day has changed, during the last year, or see if we still have the same ratio of errors now compared to, for example, half a year ago. We can also see if we still get the same types of errors as we did in past.

To be able to track this information, at first you need to create snapshots via Transaction /AIF/GENMSGSNAP. On the selection screen, you need to provide an effective date for which the report should be generated as seen in Figure 112. Using the provided date, SAP Application Interface Framework

will aggregate information about messages, their statuses, and the log messages that occurred for those interface messages.

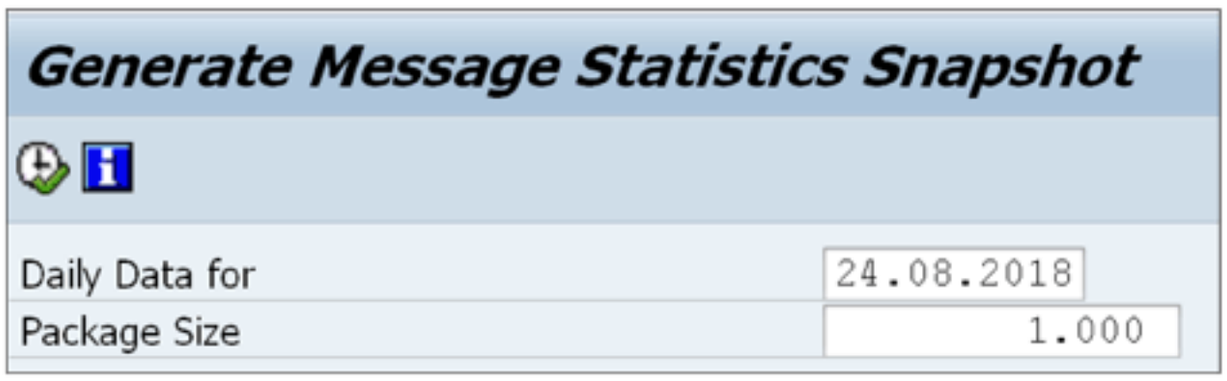

Figure 112 Snapshot Creation

The **Package Size** parameter is only for performance purposes. SAP Application Interface Framework will collect messages in packages of the specified size and will analyze them also in those packages. You can leave the parameter with its default value of 1000, unless you are experiencing performance issues with the snapshot generation. You can then use this factor to tweak the performance.

When you execute the program, a snapshot will be created, and you will be informed about a number (identifier) under which SAP Application Interface Framework stored it in the database.

Once the snapshot is generated, you can view it with Transaction /AIF/DISPMSGSNAP. On the selection screen, you can specify the following information to narrow the results:

- **Snapshot Number**
  To use if you are only interested in a specific snapshot.
- **Snapshot Creation Date**
  If you are not familiar with the snapshot number, you can always use the date of creation.
- **Snapshot Creation User**
  To select snapshots created by a particular user.
- **Snapshot Effective Date**
  This date refers to the date that was used as the **Daily Data For** parameter in the snapshot generation program.

When you execute the report, you will see a list of all selected snapshots. You can then drill down to any of them and see the following information by selecting a snapshot in the list and clicking either of the buttons listed below:

- **Cumm. Msg. Statistic**
  To see a list of all interface messages in the system up to the effective date of the snapshot (see example in Figure 113)
- **Daily Message Statistic**
  To see a list of interface messages only from the effective date of the snapshot
- **Cumm.Msg. Summary**
  To see a combined list of all log messages, aggregated by their type, existing in the system, up to the effective date of the snapshot (see example in Figure 113)
- **Daily Message Summary**
  To see a combined list of all log messages, aggregated by their type, existing in the system, only on the effective date of the snapshot

Cumulated Message Statistics

Client	SnapshotNo	NS	Interface Name	Version	NS	Recipient for Alert	All Messages	Warnings	Errors	Technical Errors	In Process	Success	Canceled	Key Fields
400	7	NOWAKM	OINVOIC02	00001	NOWAKM	DEMO_AIF	6	0	0	0	0	6	0	<fallback>
400	7	ZDEMO	IIDSORDE01	1	ZDEMO	DEFAULT_RECP	315	0	76	0	0	239	0	<fallback>
400	7	ZDEMO	IIDSORDE01	1	ZDEMO	BUS_RECP	27	0	27	0	0	0	0	<fallback>

Figure 113 Cumulated Message Statistics Example

6 Built-In Authorizations

The SAP Application Interface Framework takes advantage of several SAP authorization features to manage access and actions available to users. The principal authorization restrictions are applied using user roles. More advanced requirements can be fulfilled with the help of custom authorization objects assigned at the interface level. In addition, considering the

critical meaning of data consistency, SAP Application Interface Framework tracks any modification to message content to allow a trace back to the responsible user.

6.1 Authorization Objects and Roles

All features of SAP Application Interface Framework are based on detailed authorization objects, each serving different purposes. For example, for monitoring, the /AIF/ERR object is used. Access to SAP Application Interface Framework customizing requires the /AIF/CUST object. Furthermore, each object consists of standard SAP Application Interface Framework fields like **Namespace**, **Interface**, and **Version**, leaving much room for precisely targeted solutions. As a result, a group of users can be authorized only for certain namespaces or groups of interfaces.

Furthermore, an extensive list of available activities is provided. This allows you to, for example, grant some users display-only authorization, while others may be allowed additionally to restart and cancel messages.

The list of available activities is as follows:

- **(16) Execute**
  Selection from indexing tables
- **(33) Read**
  Access to persistence layer to read and load messages
- **(34) Write**
  Edit message content via editable fields
- **(70) Administer**
  qRFC monitoring
- **(71) Analyze**
  Access and display application log content
- **(75) Remove**
  Message cancellation

- **(A4) Resubmit**
  Message restart
- **(GL) General overview**
  XML monitoring tools
- **(24) Archive**
  Run archiving report for message persistence
- **(25) Reload**
  Load archived messages
- **(56) Display archive**
  Display archived persistence data

The drawback of this detailed authorization approach is that a correct and well-customized setup can be time-consuming to establish.

With SAP Application Interface Framework, SAP delivered a set of template roles that can be used to create actual roles according to customer needs (Figure 114). The proposed roles cover a typical set of required authorizations that would match usual user profiles:

- **SAP_AIF_ADMIN**
  User responsible for advanced tasks not related directly to development: custom functions/hints/message texts, interface determination, archiving, and correction reports.
- **AP_AIF_ALL**
  Role template with all SAP Application Interface Framework authorization objects with all activities and all SAP Application Interface Framework transactions.
- **SAP_AIF_ARCHITECT**
  User responsible for planning and coordination of SAP Application Interface Framework development.
- **SAP_AIF_DEVELOPER**
  User responsible for the development of interfaces.

- **SAP_AIF_USER**
  Dedicated to SAP Application Interface Framework business users responsible for monitoring interfaces and error handling, including, among others, restarting, canceling, and editing fields.
- **SAP_AIF_POWER_USER**
  User responsible not only for monitoring and error handling but also for advanced functions—for example, archiving, correction reports, message snapshots, scheduling file uploads from the application server, performance tracking, runtime configuration groups, defining automatic reprocessing, and configuring data transfer (e.g., qRFC interfaces).
- **SAP_AIF_PROCESSING**
  Minimal authorization for processing SAP Application Interface Framework messages; can be assigned to system users as part of message transfer.
- **SAP_AIF_TEST_TEMPL**
  Role with not only SAP Application Interface Framework authorizations and transactions but also several other authorizations and transactions that are needed for some test scenarios; not for productive use.

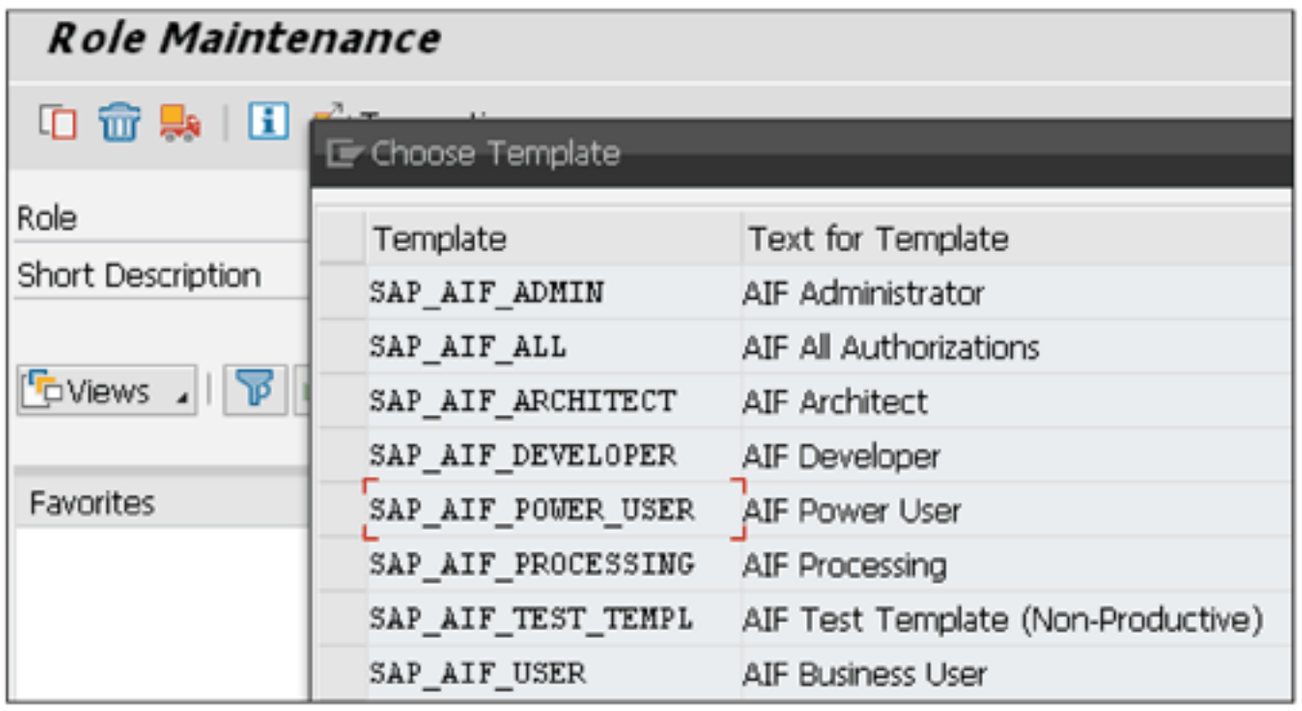

Figure 114 Transaction PFCG: Available Templates

6.2 Advanced Authorization Handling Based on Message Content

In the SAP Application Interface Framework, you can assign authorization objects to interfaces to check against the content of messages. Based on the results of such a check, a user can be granted access to certain SAP Application Interface Framework actions (e.g., message cancel). The set of available actions also can be adjusted using the same authorization object. This approach allows advance rules to be put in place to manage user access to messages and actions.

To demonstrate configuration steps, let's work through an example.

First, as shown in Figure 115 let's add a key field to our indexing table that we will check against for user authorizations. In Transaction SE11, add the BSART field to table ZAIF_T_IDX_SORDE.

Transparent Table ZAIF_T_IDX_SORDE Active

Short Description Index table for ZDEMO-IIDSORDE01-1 AIF interface

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields

Srch Help | Built-In Type | 1 / 9

Field	Key	Ini...	Data element	Data Type	Length	Deci...	Short Description
MANDT	✓	✓	MANDT	CLNT	3	0	Client
MSGGUID	✓	✓	GUID_32	CHAR	32	0	GUID in 'CHAR' Format in Uppercase
.INCLUDE			/AIF/IFKEYS	STRU	0	0	Keys
.INCLUDE			/AIF/ADMIN	STRU	0	0	Additional information structure (be appent to tran. table)
PID			SXMSPID	CHAR	40	0	Integration Engine: Pipeline ID
VBELN			VBELN_VA	CHAR	10	0	Sales Document
BSTKD			BSTKD	CHAR	35	0	Customer Reference
KUNNR			KUNAG	CHAR	10	0	Sold-to party
BSART			BSART	CHAR	4	0	Order Type (Purchasing)

Figure 115 Transaction SE11: Adding Field for Indexing and Authorization

Next, create a new authorization object. To do so, execute Transaction SU21. There we create a new object and add two authorization fields, ACTVT and BSART (Figure 116).

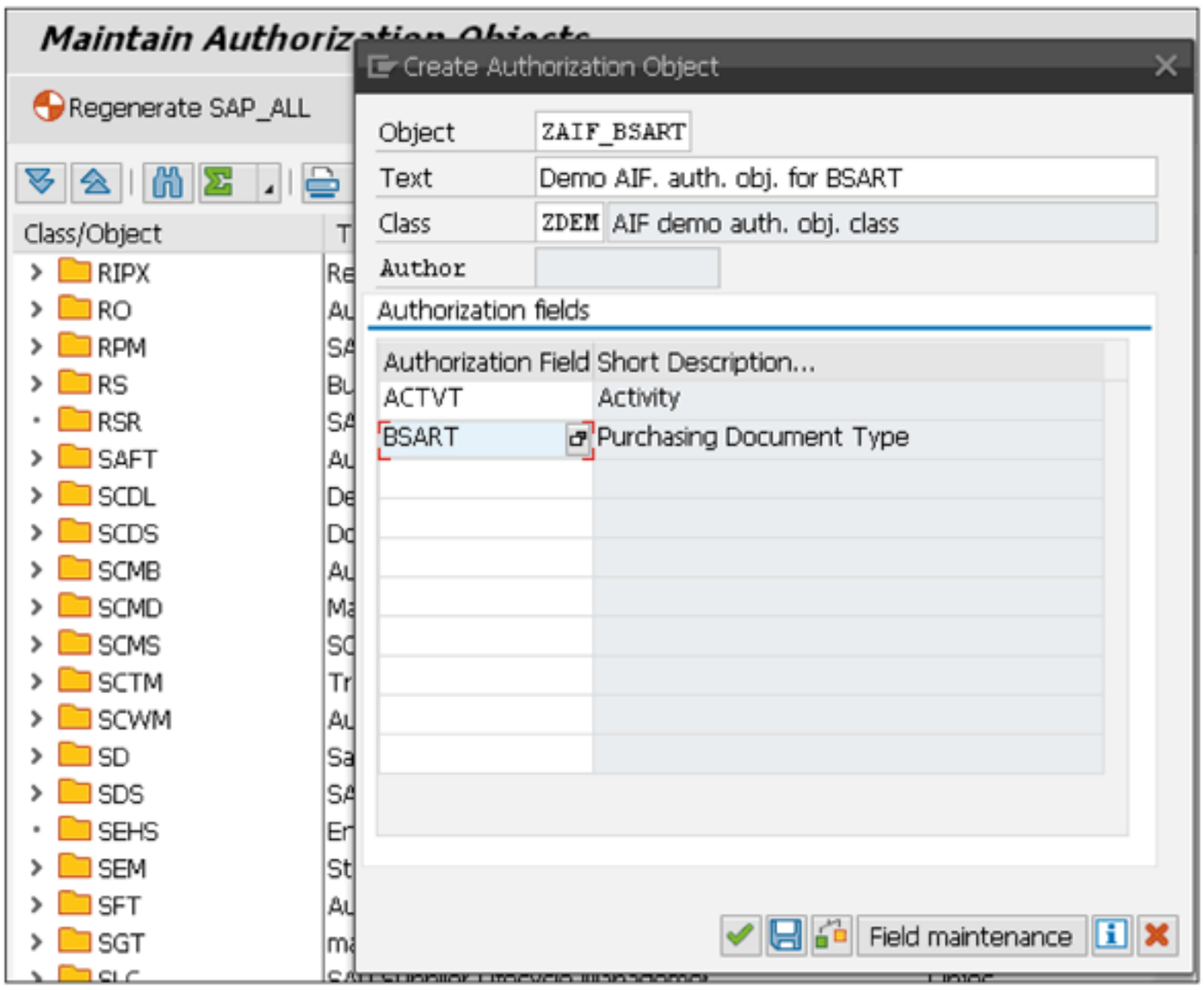

Figure 116 Transaction SU21: Creating New Auth. Object

Then from the list of **Permitted Activities** for the ACTVT field, select all the activities that are in use by the /AIF/ERR authorization object, as shown in Figure 117.

Next, open SAP Application Interface Framework customizing (Transaction /AIF/CUST) and go into **Application Interface Framework • Error Handling • Define Interface-Specific Features • Define Key Fields for Multi. Search**. There you can define a new key field for the interface using the BSART field we added in previous steps to the indexing table. The field should be populated from IDoc E1EDK01-BSART field as shown in Figure 118.

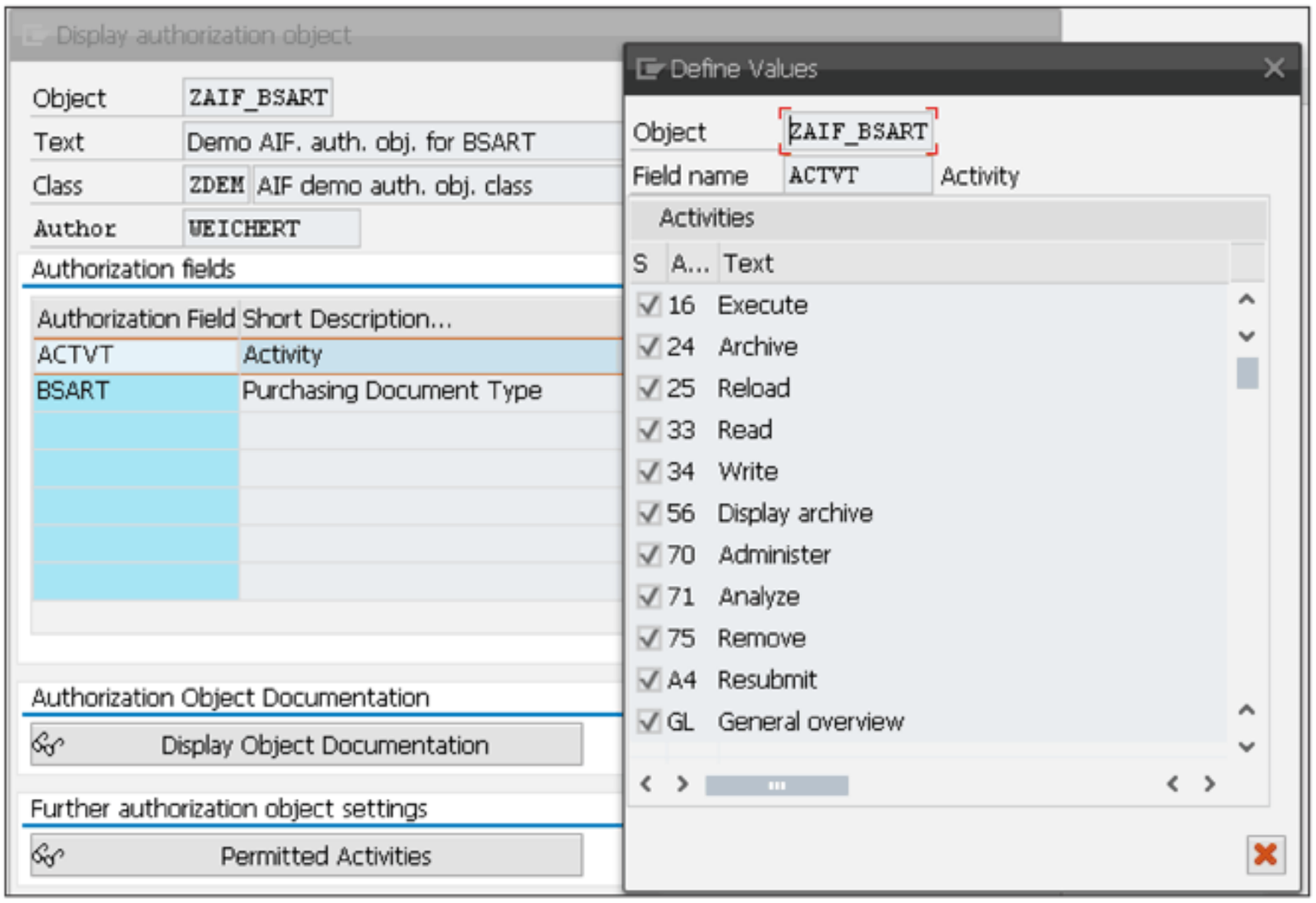

Figure 117 Transaction SU21: Maintaining Auth. Object Activities

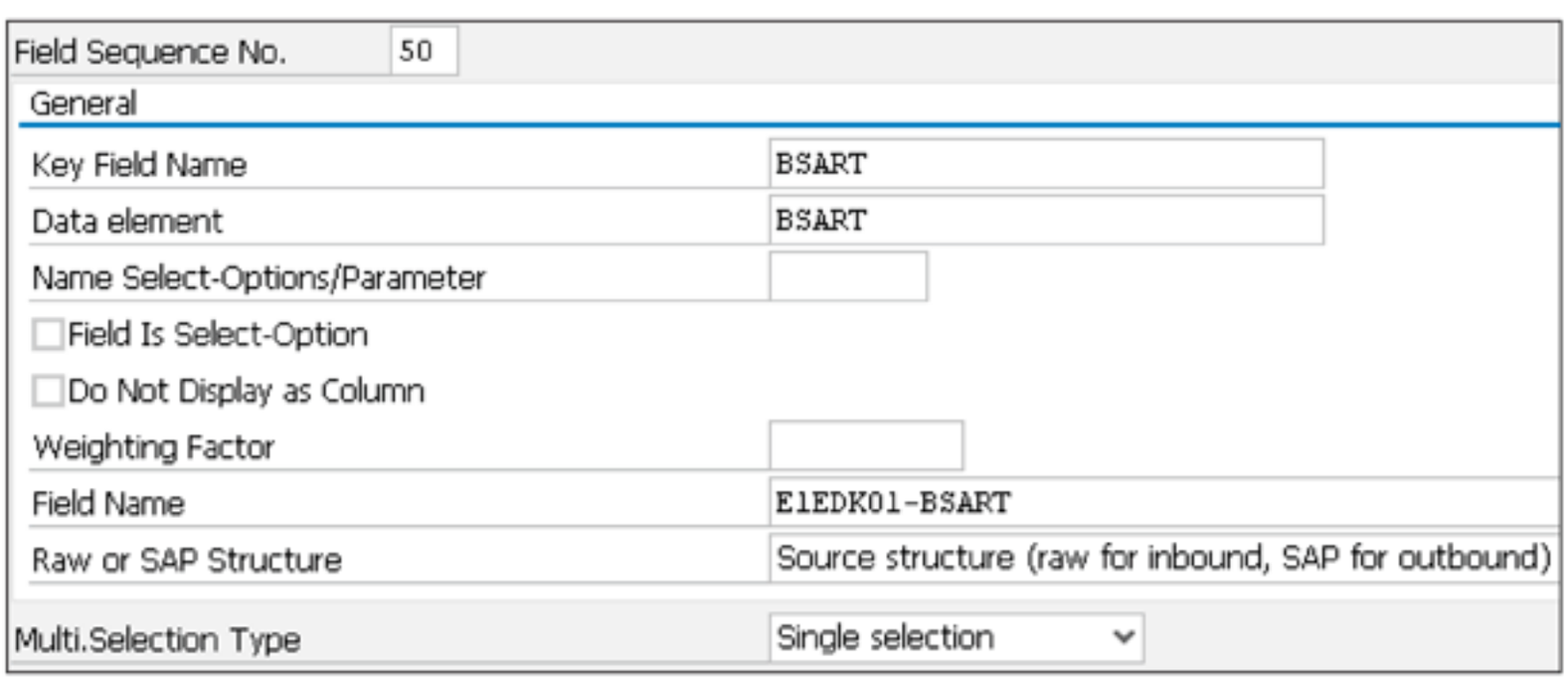

Figure 118 Define Key Field to Use for Authorization

Once you have both the authorization object and a key field to use for the check, you can assign one to the other. Still in the **Define Interface-Specific Features** customizing branch, switch to **Assign Authorization Objects to AIF Interface**. Add a new entry to the Authorization Objects table, putting in the name of our new authorization object: `ZAIF_BSART` as shown in Figure 119.

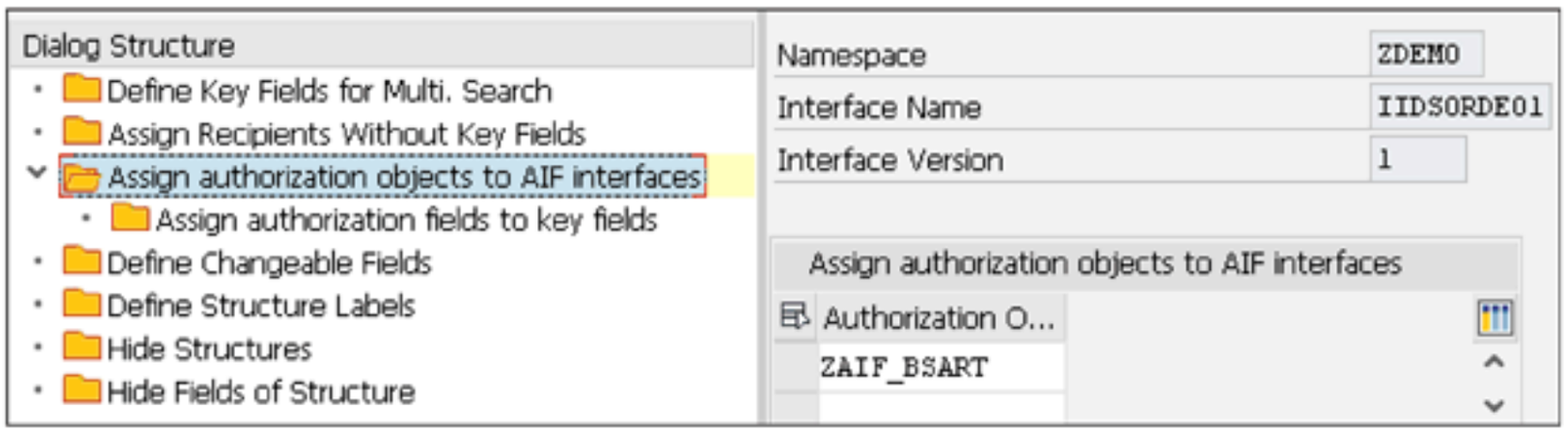

Figure 119 Assign Auth. Obj. to Interface

Next go to **Assign Authorization Fields to Key Fields** to create a new entry and populate it with the BSART key field details as in Figure 120.

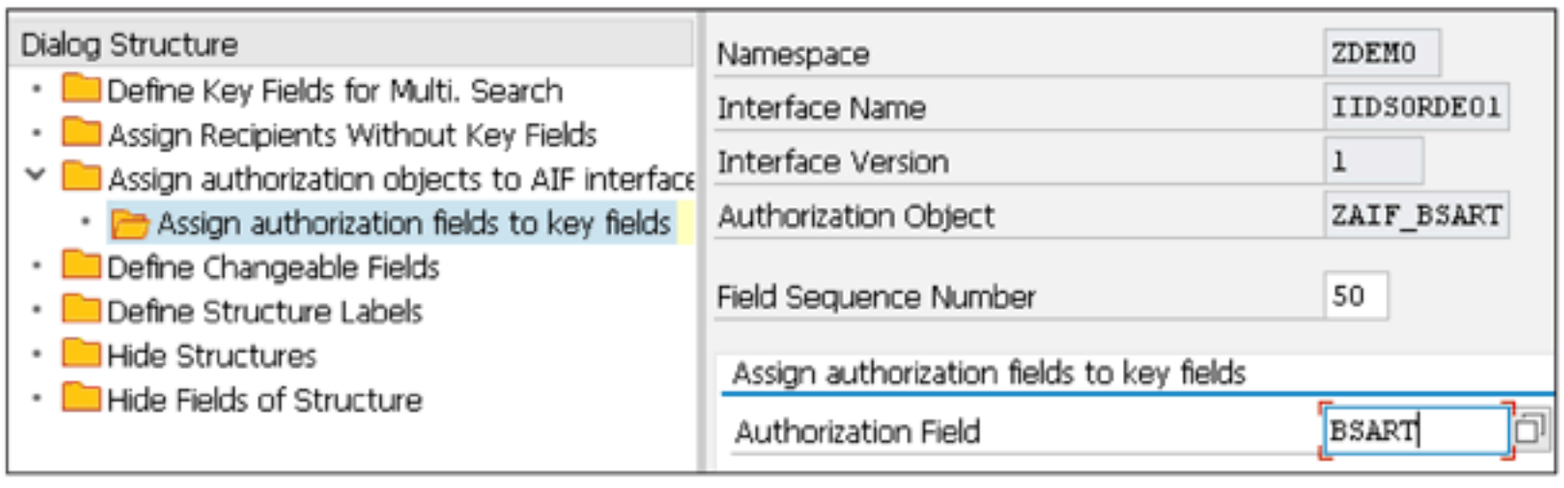

Figure 120 Assign Auth. Field to Key Field

Customizing is finished! We can now test how the authorization object affects the monitoring and error handling tool. On entering, we're immediately notified with an error message that authorization is required to display messages of the interface (Figure 121).

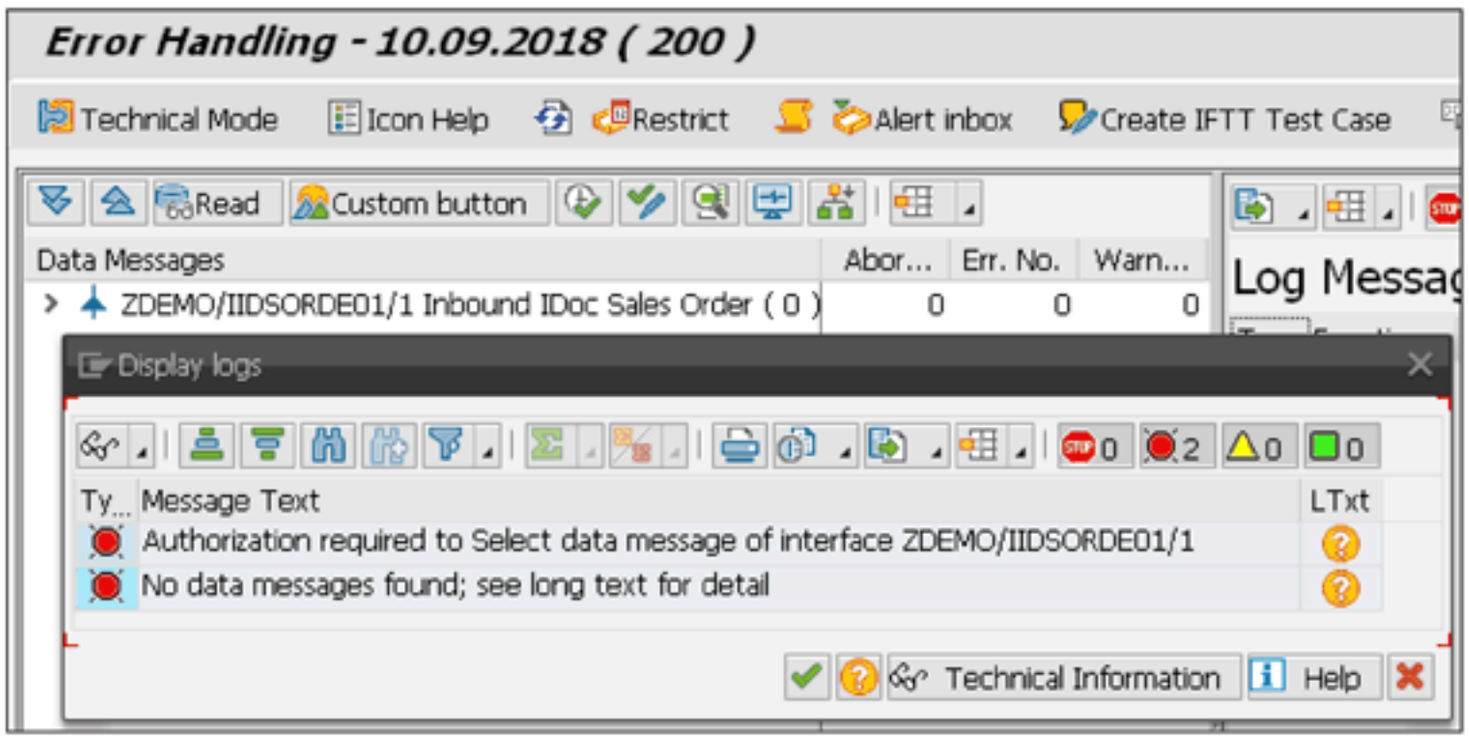

Figure 121 Transaction /AIF/ERR: Authorization Error Messsage

The error is expected because the new authorization object check is in place and we don't have it assigned in the user profile. To amend this, execute Transaction SU01, select the **Roles** tab, and from available roles select one to enhance with the new authorization object. Then change **Authorization Data** of the selected role by adding the new authorization object ZAIF_BSART. You can now specify what activities the role will now allow; let's select all of them as shown in Figure 122. Furthermore, we provide a value for the BSART field (we'll use "Z001" for our test). It will be checked against the key field (message content) to decide if the user should be authorized for actions specified in the ACTVT field.

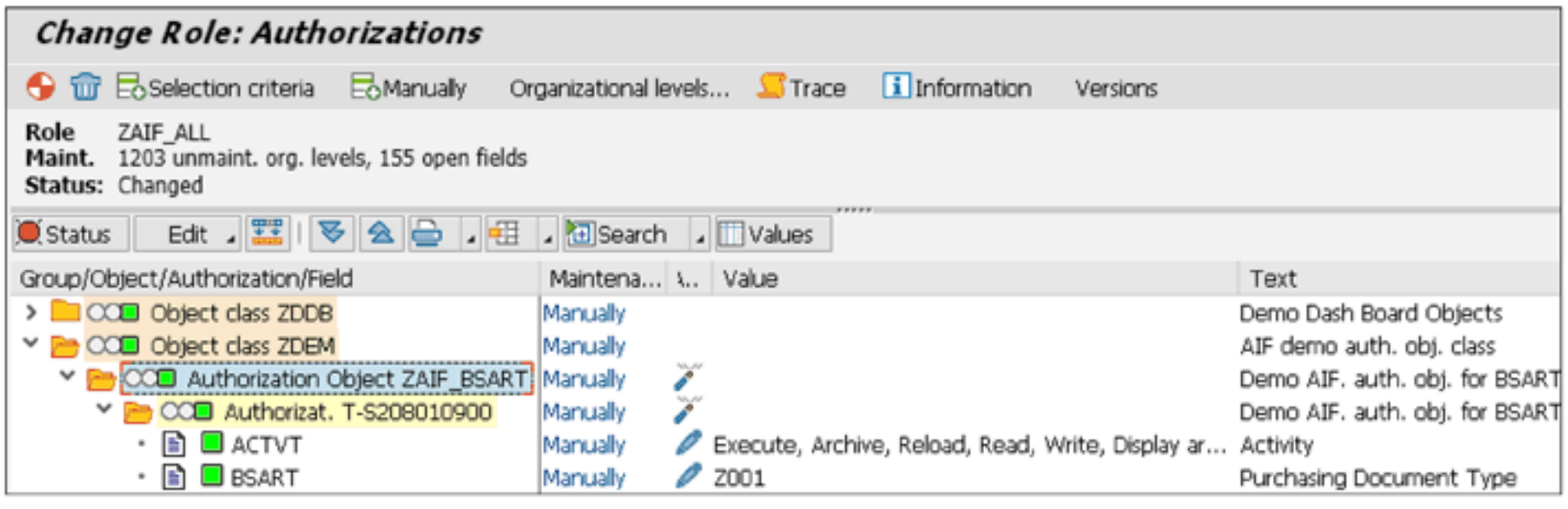

Figure 122 Role Maintenance: Adding New Auth. Object

Once done, we can try again to display data in monitoring and error handling. This time, a message is successfully displayed. Notice that E1EDK-BSART (and the related key field) content matches the value specified for the authorization object (Figure 123).

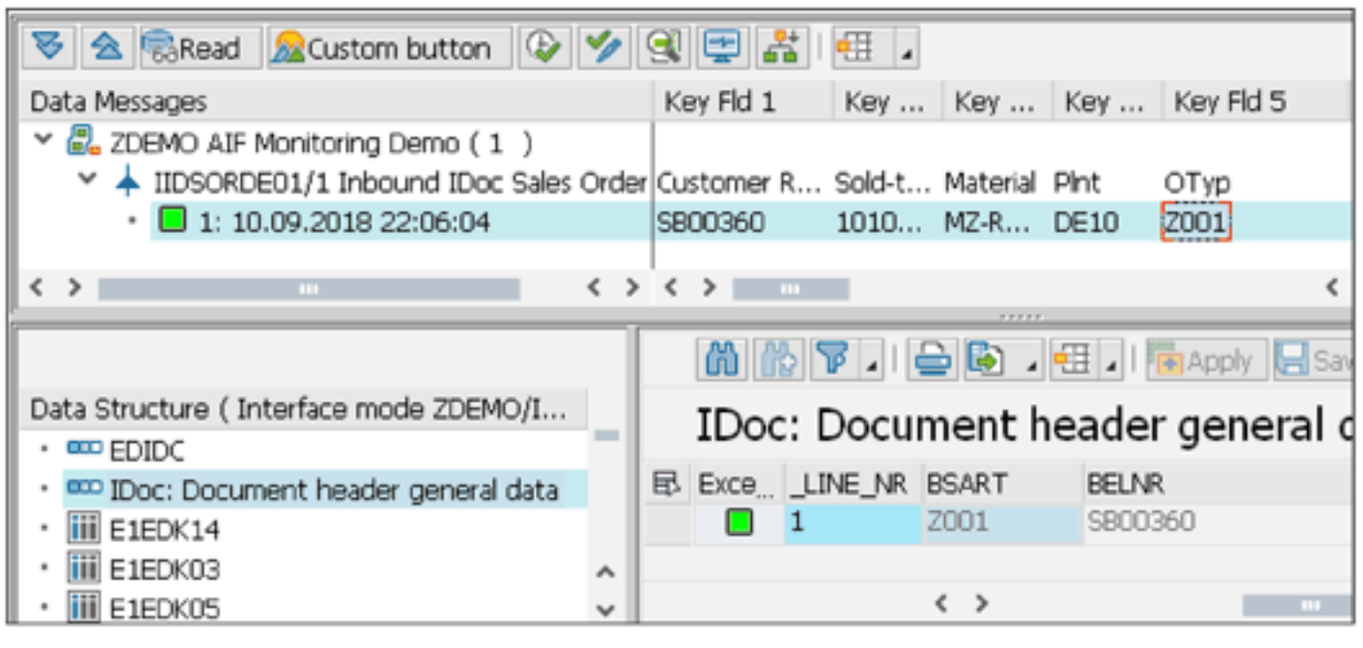

Figure 123 Transaction /AIF/ERR: Messages Matching Auth. Object Value Displayed

7 Business Process and Interface Monitoring

The current trends on the market show that IT systems are no longer supposed to only collect data and show simple information. They should now present users with extensive information, which can lead to the best actions and decisions. This, for example, means that the context of the presented information is important.

Interface monitoring tools, including SAP Application Interface Framework monitoring and error handling or SAP Application Interface Framework Interface monitor, provide users with vast knowledge about the exchanged messages. Using their capabilities, we can quickly analyze an error and find its root cause. Some, like SAP Application Interface Framework Interface monitor, can present a nice overview and give quick info on how well the interfaces are doing. But what if you want to see more? For technical teams specializing in interfaces, a message in error is just another task that needs to be resolved. But for the business, the same message can mean a big order on hold or an invoice not delivered to customer. The interfaces are always just a part, but sometimes a very significant part, of a bigger process. What if we want to see our interface messages in a context of an entire business process?

The tools described in this section can make it happen.

7.1 Monitoring with the Process Observer

Process observer is a component of the SAP Business Suite foundation layer. It is a tool that is responsible for monitoring business processes in SAP application systems, using events raised by objects from SAP Business Objects Repository (BOR), as well as non-BOR events that can be raised by SAP applications. We might not be aware of that, but these events are triggered in a majority of SAP transactions. An event can be, for example, sales order creation, goods issue, or invoice posting. Over 7,000 available BOR events cover most of the standard processes in SAP Business Suite.

Note

Process observer is available with SAP NetWeaver Business Suite Foundation 731, 702 SP06, 701 SP11, and higher releases.

Using the process observer process definition, we can define a model of our business process. This model will be a set of events that can occur during the whole lifetime of the process. Each event can progress it ahead, until the finish, or put it on hold. Having detected an event in the system, process observer will create a log entry. Each entry will belong to a specific business process instance. For example, let's assume we are monitoring an order-to-cash process. If sales orders A and B are created in the system, two different log entries for each event will fall to separate instances of a process. Next, if a delivery is created for sales order A, the log entry for that event will belong to the instance of the sales order A process. A set of log entries that correspond to the same business process instance builds a process log. And with this log, we gain great visibility into our business processes.

But how does that relate to SAP Application Interface Framework and our interfaces? Starting from release 3.0, SAP Application Interface Framework supports integration with process observer. Essentially, what it means is that SAP Application Interface Framework also can raise events that will be logged by process observer. Details of the configuration necessary to achieve this are described in the subsections ahead. In this scenario, we will configure our demo interface for IDoc ORDERS05 to raise events that will be collected by process observer. Our process will start when a new message is received in SAP Application Interface Framework and will end when a successful sales order is created.

Process Observer Configuration

A majority of the configuration that needs to be done for this scenario will be performed in process observer, outside of SAP Application Interface Framework. We are aiming at a process definition that will log the following events:

- Message arrived into SAP Application Interface Framework
- Error when processing in SAP Application Interface Framework
- Message cancelled in SAP Application Interface Framework
- Message restarted in SAP Application Interface Framework
- Message successfully processed in SAP Application Interface Framework
- Sales order created
- Sales order changed

The next sections describe the steps to achieve this.

Process Observer Activation

We need to activate process observer in our system. To do that, call Transaction SPRO and go to **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Activation • Activate Process Observer**. Select **Parameter ID** = 'POC_ACTIVE' and Type "X" as **Parameter Value**. **Save** your entry. Now select the next customizing node, **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Activation • Schedule Event Processing Job**, and click **Execute** or press F8. This will schedule a job that will run every one minute (by default) and will process any events that were logged since the last job run and for which a valid process definition in process observer exists.

Process Facade Definition

As a next step we will build a facade layer for our process. The facade defines the core object and events that will be used in our process definition. Follow these steps to create it:

1. Go to **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Facade Layer Content • Maintain Objects in Facade Layer**. First, create a ZAIF object, which will serve as a generic object for all SAP Application Interface

Framework–related events. Double-click the **SOA BO Types** node and create a new entry, ZAIF, as shown in Figure 124.

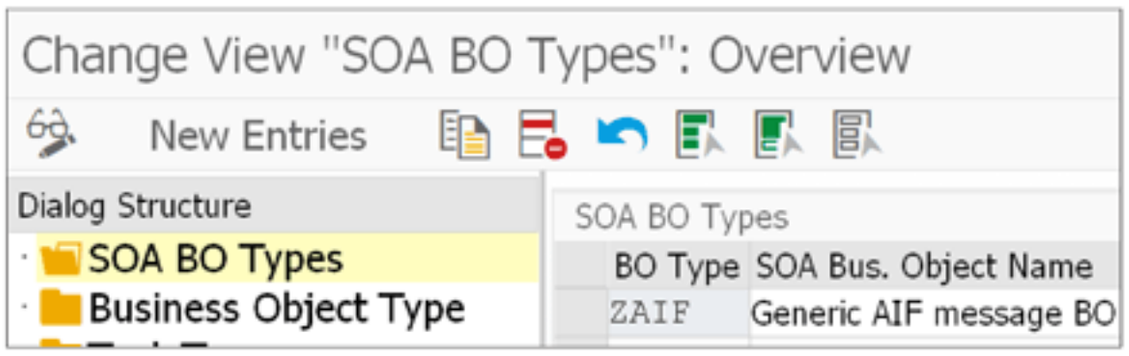

Figure 124 New SOA BO Type Definition

2. Next, double-click the **Business Object Type** node and create a similar entry. Maintain "ZAIF" as the **BO Type** and **Generic AIF Message** as the **Business Object**.

3. Now, with the core business object defined, we need to define tasks for our business object. The tasks correspond to different events that can occur throughout the process. In our scenario, we will be reusing some standard, existing BOR events for sales order creation and sales order change. That means that for business object ZAIF, we only need to set up tasks that are related to SAP Application Interface Framework events. To do that, double-click the **Task Type** node and define the entries shown in Table 3.

Task Type ID	Task Type
Z010	Arrived in AIF
Z011	Restarted in AIF
Z020	Error in AIF
Z025	Cancelled in AIF
Z030	Finished in AIF

Table 3 Task Types for New Business Object ZAIF

4. Next, assign the already created task types to custom business object ZAIF. Double-click the **Task** node and define entries, as shown in Figure 125.

Task

BO Type	TaskTy.ID	Task
ZAIF	Z010	Arrived in AIF
ZAIF	Z011	Restarted in AIF
ZAIF	Z020	Error in AIF
ZAIF	Z025	Cancelled in AIF
ZAIF	Z030	Finished in AIF

Figure 125 Assign Task Types to Business Object ZAIF

5. Now, for custom object ZAIF, assign a BOR object. We will use object IDOCORDERS, which is already defined in the system as part of SAP standard. This BOR refers to IDoc ORDERS, which we use in this exercise.

   Choose path **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Application Instrumentation • BOR Event Instrumentation • Map BOR Events to Tasks**. Maintain an entry as presented in Figure 126.

Dialog Structure
- Map Business Object Type To BOR Object
- Map BOR Event to Task

Map Business Object Type To BOR Object

BO Type	Business Object	Object type	Description
ZAIF	Generic AIF message	IDOCORDERS	IDoc message ORDERS

Figure 126 Mapping Object ZAIF to IDOCORDERS

6. **Save** your entry and exit.

In the next section, using the facade objects we have created in this step, we will prepare a process definition.

Process Definition

With the process facade ready, let's continue to the process observer process definition steps:

1. Proceed to **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Process Definition • Create and Maintain Process Definition**.
2. In the **Process Definition Header** node, make a new entry:
   - **Process Definition ID**: IDOC_TO_ORDER

- **Process Definition**: from IDoc to Sales Order
- **Log Level**: 2, standard logging

3. Select the new entry, and double-click the **Process Version** node. Create a new entry with the following values:
   - **Version**: 1
   - **Active**: checked
4. Next, select the entry, and double click on node **Activities**. In this step we will model our process, using the tasks that we defined in previous steps, as well as the tasks that we will reuse from standard objects. See Figure 127 for a reference of the activities that need to be set up in our process definition.

Process Def. ID: IDOC_TO_ORDER
Process Version: 1

Activities

Acty. ID	Activity	Start	End
ARR AIF	Arrived in AIF	☑	☐
CHG SOR	Change Sales Order	☐	☐
CNC AIF	Cancelled in AIF	☐	☑
CRE SOR	Sales Order Created	☐	☐
ERR AIF	Error in AIF	☐	☐
FIN AIF	Finished in AIF	☐	☑
RES AIF	Restarted in AIF	☐	☐

Figure 127 Activities of Process IDOC_TO_ORDER

5. The activity ARR AIF in the process definition is marked as **Start Activity**, and activities CNC AIF and FIN AIF are marked as **End Activities**. This means that process observer will treat the ARR AIF activity as a starting activity for a new process instance, whereas CNC AIF (cancelling SAP Application Interface Framework message) and FIN AIF (finish processing of message in SAP Application Interface Framework) will end the process.
6. In addition, we need to assign actual events defined in previous steps to the activities just defined for this process definition. To do that, for each of the defined activities, select it and double-click the **Task Assignment** node.

7. Make entries as presented in Table 4.

Activity ID	BO Type	Task Type
ARR AIF	ZAIF	Z010
RES AIF	ZAIF	Z011
ERR AIF	ZAIF	Z020
CNC AIF	ZAIF	Z025
FIN AIF	ZAIF	Z030
CRE SOR	114	21
CHG SOR	114	88

Table 4 Mapping of Activity IDs to Task Types

Note

For all activities except CRE SOR and CHG SOR, we are using the custom-defined object ZAIF. For the remaining activities, standard object 114 and corresponding Task Types 21 and 88 are used. This is thanks to the standard functionality provided for a majority of SAP BOR objects, such as sales order.

In the next step we will be activating process logging for our new process definition.

1. Proceed to path **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Process Logging • Activate Process Definition Logging**, and make an entry as shown in Figure 128.

Process Definition: Header Details

Process Definition ID	Process Definition	Current Version	Business Area	Log Level
IDOC_TO_ORDER	From IDoc to Sales Order	1		2 Standard logging

Figure 128 Activating Logging for Process Definition

2. Next, go to **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Process**

Definition • Check Process Definition and execute the check for your process definition IDOC_TO_ORDER, with the **Show Active Only** and **Additional Checks** options selected. Make sure the check results are as presented in Figure 129.

Status of Process Definitions

Process Definition

Process Type	Description	Consistency	Key
Process Definitions	Show active only		
From IDoc to Sales Order	Active	✓	IDOC_TO_ORDER
Current version	Active	✓	001
Arrived in AIF	No BOR event or class assigned to some tasks	!	ARR AIF
Cancelled in AIF	No BOR event or class assigned to some tasks	!	CNC AIF
Error in AIF	No BOR event or class assigned to some tasks	!	ERR AIF
Finished in AIF	No BOR event or class assigned to some tasks	!	FIN AIF
Restarted in AIF	No BOR event or class assigned to some tasks	!	RES AIF
No sequence information at all			
Scheduling	Scheduled	✓	POCR_MAIN_QUEUE
Activation	Active	✓	POC_ACTIVE
Registry for Federation	Local		

Figure 129 Status Check of Process Definition

You can ignore the warnings for the custom tasks defined for the ZAIF business object, as they are not critical for our scenario.

Creating a Link between Standard Events and SAP Application Interface Framework–Related Events

As you may have noticed, we are using two different business objects in our process definition. The ZAIF business object is our custom object that covers all SAP Application Interface Framework–related events. The 114 business object is a standard object provided by SAP for sales order–related events. Because we want both objects working together in one process definition, we need to enable a link between them. A link will be created between the **Sales Order Create** event (Activity **CRE SOR**) and **Processed in AIF** event (Activity **FIN AIF**). Whenever a **Sales Order Create** event is raised, it will look for a corresponding **Finish in AIF** event from the same process instance.

How can this be achieved? Every business object instance has its own object key. In our example, for the ZAIF business object, the object key will be the IDoc number (which is also the SAP Application Interface Framework

message GUID). For the 114 business object, the object key will be the sales order number.

What we need to do is make the first event of the 114 sales order business object (**Sales Order Created**) to point to the last event of the ZAIF business object (**Finished in AIF**). The way they can be linked together is by using the **Customer Purchase Order Number** field, which is stored in both the IDoc message (field **E1EDK02-BELNR** for qualifier **001**) and the sales order document in SAP (field **VBKD-BSTKD**). Follow the steps ahead to implement this linking.

Implementation of Necessary BAdIs

To create this link, we will use one of the multiple BADIs provided by process observer. Here is how to do so:

1. The first BAdI we will use is called whenever an event is raised and captured by process observer. When executing it, we can determine a link to a previous business object that already exists as part of an open process instance. To implement the BAdI, go to **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Application Instrumentation • BOR Event Instrumentation • BAdI: Determination of Previous Business Object for BOR Event**. Click **Create Implementation** to create a new BAdI implementation (Figure 130).

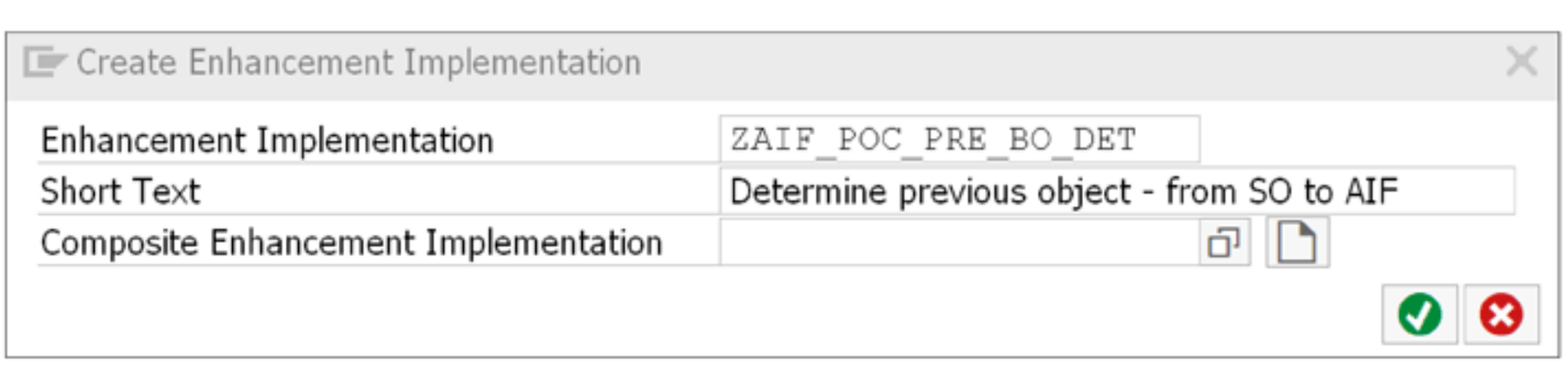

Figure 130 Create New BAdI Implementaion

2. On the next screen, specify the package and transport number. Finally, maintain the following fields:
   - **BADI Implementation Name** as "ZAIF_POC_PRE_BO_DET_ENH"
   - **Implementation Class** as "ZCL_POC_INSTR_PRE_BO_DET"

   If you are asked to decide if you want to reuse the existing sample implementation of the BAdI, choose options to **Create Empty Implementation** and use **Empty Class**.
3. Now go to the implementing class, `ZCL_POC_INSTR_PRE_BO_DET`, and implement the `IF_POC_PRE_BO_DET~DETERMINE_PRE_BO` method, using the code snippet in Listing 5.

```
METHOD if_poc_pre_bo_det~determine_pre_bo.
*  ***********************************************************************
*  The structure IS_EVENT is an event for which the pre object has
*  to be determined
*  ET_PREVIOUS_BOR_OBJECTS should be filled with table of previous
*  objects
*  ***********************************************************************
    DATA: ls_borident TYPE borident,
          lv_belnr    TYPE bstkd,
          lv_msgguid  TYPE guid_32,
          ls_vbkd     TYPE vbkd,
          lv_vbeln    TYPE vbkd-vbeln.
    IF is_event-objtype = 'BUS2032'.
      lv_vbeln = is_event-objkey.
      CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
        EXPORTING
          input  = lv_vbeln
        IMPORTING
          output = lv_vbeln.
      SELECT SINGLE bstkd FROM vbkd INTO lv_belnr
        WHERE vbeln = lv_vbeln.
IF sy-subrc = 0.
SELECT SINGLE msgguid FROM ZAIF_T_IDX_SORDE INTO lv_msgguid
          WHERE bstkd = lv_belnr.
 IF sy-subrc = 0.
          ls_borident-objtype = 'IDOCORDERS'.
```

```
          ls_boridٍent-objkey = lv_msgguid.
          APPEND ls_boridٍent TO et_previous_bor_objects.
 ENDIF.
ENDIF.
ENDIF.

ENDMETHOD.
```

Listing 5 BAdI Implementation ZAIF_POC_PRE_BO_DET_ENH

4. This code will be executed every time an event is raised to process observer. In the first step, it checks if the BOR type is BUS2032 (which is a standard BOR for sales order and assigned to process observer business object 114, which we already know). Then it will check the value of Customer Purchase Order Number (VBKD-BSTDK) and will try to find the same value in the SAP Application Interface Framework index table ZAIF_T_IDX_SORDE. This table is assigned to our demo interface ZDEMO/ IIDSORDE01/1, and the field BSTDK is one of the fields that are indexed. This means that whenever we receive an IDoc through this interface, the same value should be store in the tables ZAIF_T_IDX_SORDE-BSTDK and VBKD-BSTDK. And we will be using this link to combine the business object instances of ZAIF and 114 into one common process instance.
5. Remember to **Save** and **Activate** the code and the BAdI implementation.

Deactivating or Reimplementing a Standard BAdI

If you are using SAP S/4HANA, make sure to deactivate any default BAdI implementations visible at path **Cross-Application Components • Processes and Tools for Enterprise Applications • Process Orchestration for Built-In Processes • Process Definition • BAdI: Rule-Based Binding of Tasks to Process Activities**.

If you find the implementation presented in Figure 131 active, be sure to deactivate it.

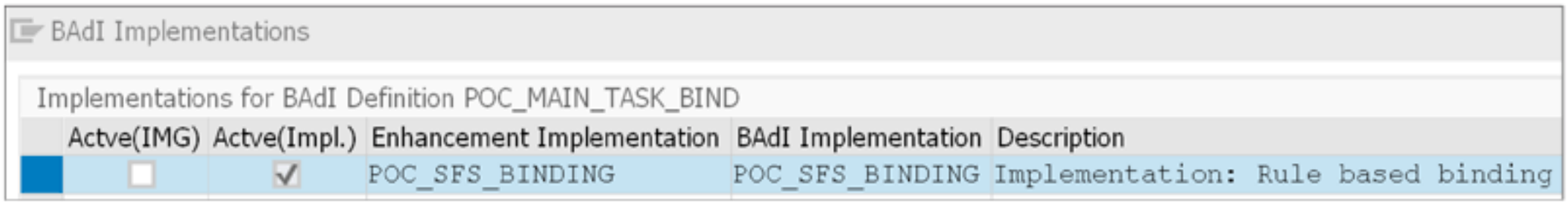

Figure 131 Deactivate POC_SFS_BINDING Implementation

To deactivate it, uncheck the **Active (IMG)** field. This implementation is part of the SAP S/4HANA order-to-cash monitoring feature. If you would like to keep this functionality active as well, make sure that it will not execute for our scenario. You can copy this implementation to a custom one; just make sure that it is not executed for process type ID IDOC_TO_ORDER.

You can add the following line from Listing 6 at the beginning of the method:

```
IF_POC_PROCESS_TASK_BINDING~TASK_TO_ACTIVITY_BIND_PRE_BO.

check cs_task_act_proc_in_bind-process_type_id <> 'IDOC_TO_
ORDER'. "prevent execution for process ID 'IDOC_TO_ORDER'.
```

Listing 6 Code to Prevent Execution of Standard Implementation POC_SFS_BINDING

This concludes the process observer part of the configuration.

SAP Application Interface Framework Configuration

The setup in SAP Application Interface Framework is much simpler. We need to assign the custom object type ZAIF and the corresponding task types to the events that can be raised by SAP Application Interface Framework. The main customizing node to perform these tasks is available once you call Transaction /AIF/CUST and proceed to the **SAP Application Interface Framework • System Configuration • Configure Interfaces for Process Observer** path. Select the namespace of your interface and double-click the **Configure Process Observer** node.

Make an entry for your interface as per the Figure 132.

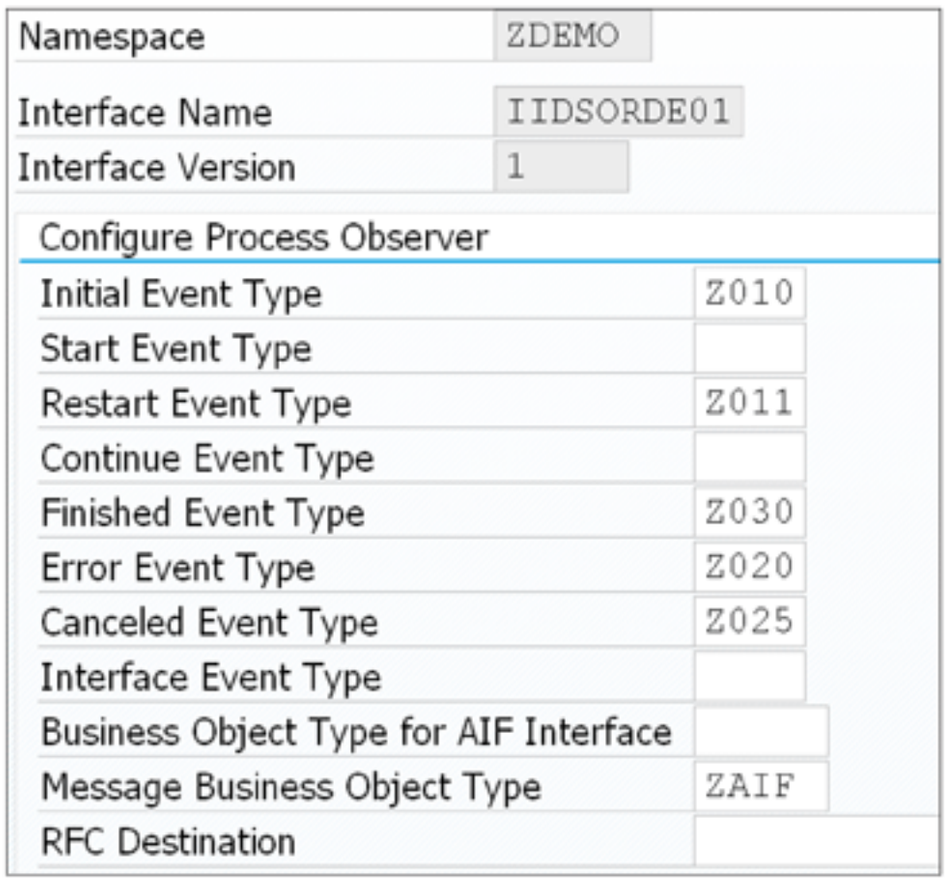

Figure 132 Configure Process Observer for SAP Application Interface Framework Interface

This customizing screen allows for assigning different task types of process observer to the actual events that can be called by SAP Application Interface Framework. Because we have defined only five different events for our custom object ZAIF, we are only assigning these tasks to our interface.

Now double-click the **Configure Interfaces for Process Observer** node and make an entry as shown in Figure 133.

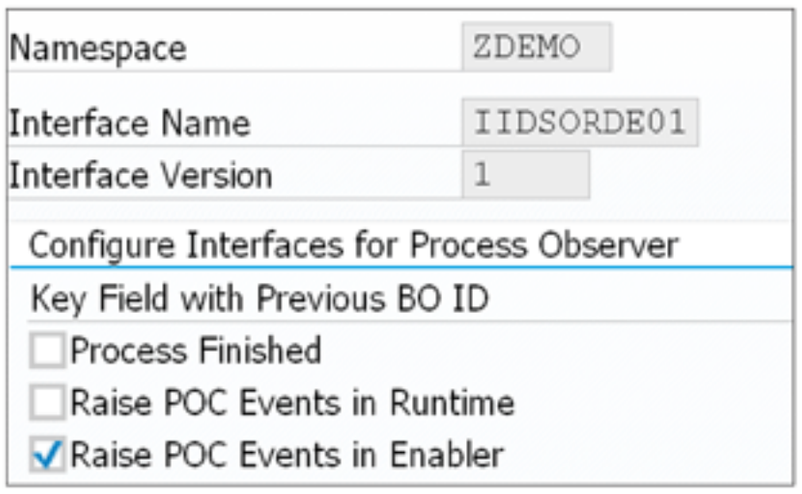

Figure 133 Configure Process Observer Properties for SAP Application Interface Framework Interface

On this screen, we are only selecting the **Raise POC Events in Enabler** option because we are only using the enabler in our scenario; we are not processing the ORDERS05 IDoc through the runtime of SAP Application Interface

Framework, but just letting SAP Application Interface Framework monitor the standard IDoc using the enabler.

This concludes the configuration necessary for our scenario.

Test

To test the process observer setup, we will send another ORDERS05 IDoc through the interface using Transaction WE19 (Figure 134).

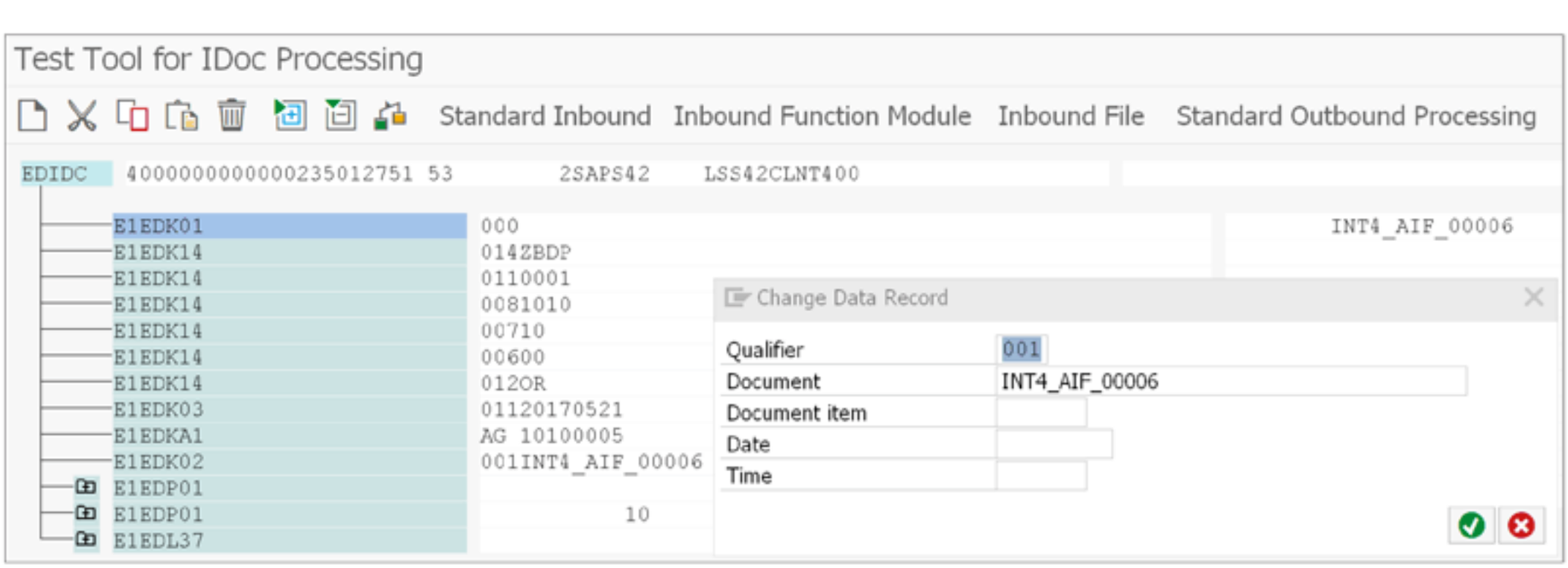

Figure 134 Test IDoc ORDERS/ORDERS05

Make sure to maintain a unique value in field **E1EDK02-BELNR** for qualifier **001** to make the process instance unique. Maintain the same value for field E1EDK02-BELNR.

Send the test IDoc using the **Standard Inbound** option. Once the IDoc is processed, go to Transaction POC_MONITOR and select **IDOC_TO_ORDER** for **Process Definition ID**. Wait about one minute, until the batch job processes the data, and press `Enter` to refresh.

You should see three events logged by process observer, as per Figure 135. Two events, **Arrived in AIF** and **Finished in AIF**, were raised by SAP Application Interface Framework interface `ZDEMO/ IIDSORDE01/1` using custom business object `ZAIF`, while the third event, **Sales Order Created**, was raised by the standard BOR object for sales order (business object type 114/BOR BUS2032). These events are tightly connected together in one business process

instance thanks to the BAdI implementation ZAIF_POC_PRE_BO_DET_ENH, defined in one of the preceding steps.

Process Details

Refresh | Process Definition | Instance Diagram

Process

Process Definition	From IDoc to Sales Order	Process Version	1
Start Date/Time	28.04.2018 03:11:23	Realized Process Chain Definition	
Start Task	Arrived in AIF	Business Object ID	0000000000235029
Process Status	Running	Threshold	

Process log (3) | KPIs (0)

Process Details

Execution Date	Execution Time	Activity	Compound Activity	Business Object ID	Executed in System
28.04.2018	03:11:24	Finished in AIF	☐	0000000000235029	Local
28.04.2018	03:11:24	Sales Order Created	☐	0000044150	Local
28.04.2018	03:11:23	Arrived in AIF	☐	0000000000235029	Local

Figure 135 POC_MONITOR Showing Process Logs from Message and Sales Order Creation

Now that our SAP Application Interface Framework interface can trigger events that will be collected by process observer, we can use some of the features provided by process observer, like graphical process visualization or KPI measurements. The following measurements are supported:

- **Count KPIs**
  Measuring the number of times the event occurred—for example, to see how many IDocs we received in the system and how many actually ended in order creation
- **Duration KPIs**
  Measuring the time between specific events—to see how much time it takes from the moment when an IDoc arrives in the system to successful sales order creation (or if we continue with the process, until an invoice is created)
- **Classification KPIs**
  To measure other KPIs using advanced BRFplus rules

For each of the KPIs, we can assign thresholds that, when exceeded, can trigger workflows or alerts. This is a much broader topic, however, which we will not cover in this E-Bite.

In the next section, we will look closer at monitoring SAP Application Interface Framework interfaces from SAP Solution Manager.

7.2 Monitoring with SAP Solution Manager

With SAP Application Interface Framework 3.0 SP 7, you have the ability to integrate SAP Application Interface Framework with SAP Solution Manager. Your SAP Solution Manager installation needs to be at least version 7.2 SP 5, for which a new integration scenario for SAP Application Interface Framework is available. Same as for your process integration scenarios, where you can see statistics on your SAP PI interfaces, here you can have SAP Solution Manager extract the main statistics on your interfaces from your SAP Application Interface Framework installation.

SAP Solution Manager is fed by the same data as SAP Application Interface Framework's Interface Monitor transaction and is part of the integration scenarios in Transaction SM_WORKCENTER. This means that the main information that you will see in SAP Solution Manager will be a status overview of your interfaces, with the following information in scope:

- Overall number of messages
- Number of errors
- Number of warnings
- Number of successful messages
- Number of canceled messages
- Number of messages in progress

Based on those numbers, you can also create metrics and assign thresholds, which, when exceeded, will trigger alerts in SAP Solution Manager.

All these figures can be presented in both namespace- and interface-specific aspects. This is a different kind of information from that provided in Process Observer monitoring. There, you can go down to a single instance of a process and see the status of particular transaction. SAP Solution Manager focuses more on a global overview of interface statuses. You will quickly see if there are any critical issues with your interfaces, but you will not see the status of a particular message. To see it and analyze it, you need to login to the respective SAP application system and use SAP Application Interface Framework monitoring to analyze it.

There are prerequisites to setup monitoring of SAP Application Interface Framework interfaces within SAP Solution Manager. First, your SAP application system with SAP Application Interface Framework needs to be connected to SAP Solution Manager and have the SAP Solution Manager agent installed.

Note

For details on basic SAP Solution Manager configuration and application system connectivity, visit http://help.sap.com/solutionmanager.

Second, in your SAP Application Interface Framework solution, you need to assign the technical user used by SAP Solution Manager to connect to SAP Application Interface Framework to the **ALL** recipient. This is a default SAP Application Interface Framework recipient that can collect all the messages from all interfaces. It is created in a blank namespace. The only action necessary is to assign the technical user used by SAP Solution Manager to this recipient under **SAP Application Interface Framework • System Configuration • Assign Recipients**. See Figure 136 for reference.

Note

Make sure you have SAP Note 2379798 installed in your SAP application system with SAP Application Interface Framework. With this SAP Note installed, you'll have a new view available, /AIF/ACT_REC_ALL, in which you can activate recipient **ALL**.

Figure 136 SAP Solution Manager User Assigned to ALL Recipient

From the SAP Solution Manager side, go to Transaction SOLMAN_SETUP. And from the **Scenarios** pane, select **Application Operations • Integration Monitoring • Interfaces and Connections**. Next, go directly to step 4, **Define Scope**, and go to **Edit** mode. Now **Create** a new scenario with the following settings:

- **Name**
  Enter "AIF_DEMO"
- **Description**
  Enter "Demo AIF setup"

In **Define Technical Systems**, assign your SAP Application Interface Framework application system. In our demo, we are focusing on the monitoring and inbound sale order interface that uses the ORDERS IDoc; you can also assign the source system that sends the IDoc to the SAP application system, such as SAP PO or SAP Cloud Platform Integration.

In the next step, **Preparation**, execute all automatic preparation steps. Next, in **Configuration**, create a new channel of type **SAP Application Interface Framework**, as shown in Figure 137. This is the new channel type, available in the new SAP Solution Manager release.

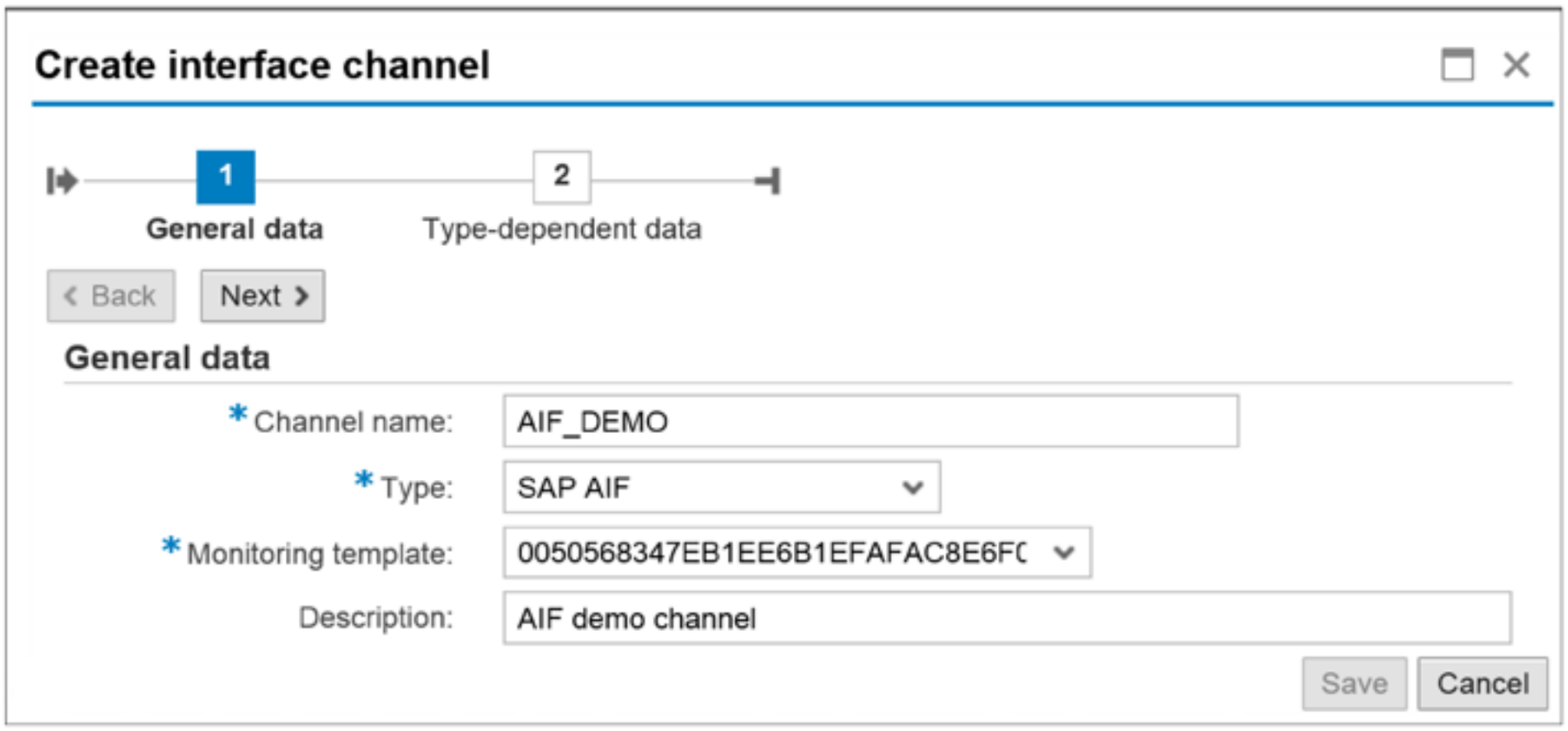

Figure 137 Defninition of New Channel of Type SAP Application Interface Framework

In the **Type-Dependent Data** tab, you need to specify your SAP Application Interface Framework system as the target system and set the source system that feeds the IDocs into the SAP Application Interface Framework system as the source.

Once the channel is created, you can select it from the list and scroll down to the **Details** of the interface channel. Here you can specify all SAP Application Interface Framework interfaces that you want to focus on by maintaining an SAP Application Interface Framework namespace or specific interface. In the next step, **Activation**, in the **Interface Channel Activation** section, for each object type you will find in the list, specify the following in the **Data Collection** tab:

- **Collection Interval**
  This parameter defines how often SAP Solution Manager should contact the SAP application system to extract data.
- **Threshold**
  This parameter defines critical values that, when exceeded, will raise a proper flag (e.g., raise red flag when more than 100 errors are detected in an hour).

Save and **Apply and Activate** your changes. In the next step, **Complete**, choose **Finish**. This concludes the setup of monitoring in SAP Solution Manager. From now on, in Transaction SM_WORKCENTER, under the **System and Application Monitoring • Interface Monitoring • Technical scenarios** path, you can select your SAP Application Interface Framework scenario and can see a status overview for all your interfaces.

Using this method, you can see almost immediately if anything that needs your attention is happening in your SAP Application Interface Framework interfaces. If you have a sudden peak or errors, you will see a proper flag raised by SAP Solution Manager and can react quickly. Compared to the monitoring options provided by process observer, these options focus on the overall performance of the interfaces, without going into the details and drilling down to specific process instances.

8 Summary

Implementation of a new solution like SAP Application Interface Framework is always a challenge. That is why it might be worthwhile to focus, during the first wave of SAP Application Interface Framework implementation, on the SAP Application Interface Framework monitoring options for your interfaces, without reengineering any working solutions. This is a good technique, especially for mature environments that already have stable working interfaces. This way you get a "quick win" with lower risk by providing users with the monitoring features of SAP Application Interface Framework; then you can start working on a full SAP Application Interface Framework implementation from there to use its powerful interface design capabilities.

Taking into consideration the fact that SAP Application Interface Framework is now part of the standard SAP S/4HANA delivery, we are sure that this tool will be much more popular in years ahead. Maybe you reached for

this book because your company or your customer has decided to implement it. Having read this book, you should now have comprehensive knowledge of the monitoring features of SAP Application Interface Framework. We hope this will help you in your current or future SAP Application Interface Framework implementations.

9 What's Next?

In this E-Bite, we've walked through SAP Application Interface Framework, alerts management, statistical reports, and built-in authorizations. Now that you've seen how to handle errors and monitor interfaces with Transactions /AIF/ERR and /AIF/IFMON, what's next? See how to maintain your entire SAP landscape with SAP Solution Manager 7.2! Get the hands-on guide you need to SolMan and find everything you need to manage business applications.

In addition to this book, our editors picked a few other SAP PRESS publications that you might also be interested in. Check out the next page to learn more!

More from SAP PRESS

SAP Process Orchestration: Learn to use the AEX to configure the System Landscape Directory, work with the ES repository, and manage the integration directory in SAP PO. Build integration flows, create an SAP BPM process, and get the most out of SAP BRM.

908 pages, 2nd edition, pub. 09/2017
E-book: $69.99 | **Print:** $79.95 | **Bundle:** $89.99
www.sap-press.com/4431

SAP Cloud Platform Integration: Connect your cloud and on-premise applications! This comprehensive guide to SAP Cloud Platform Integration will teach you how to integrate processes and data in your system by developing and configuring integration flows.

791 pages, 2nd edition, pub. 08/2018
E-book: $79.99 | **Print:** $89.95 | **Bundle:** $99.99
www.sap-press.com/4650

Architecting EDI with SAP IDocs: This extensive case study showcases the requirements, standards, and capabilities you'll encounter as you build an SAP EDI system and optimize electronic information exchange between businesses via IDocs.

910 pages, 2nd edition, pub. 10/2013
E-book: $69.99 | **Print:** $79.95 | **Bundle:** $89.99
www.sap-press.com/3359

SAP PRESS E-Bites

SAP PRESS E-Bites provide you with a high-quality response to your specific project need. If you're looking for detailed instructions on a specific task; or if you need to become familiar with a small, but crucial sub-component of an SAP product; or if you want to understand all the hype around product xyz: SAP PRESS E-Bites have you covered. Authored by the top professionals in the SAP universe, E-Bites provide the excellence you know from SAP PRESS, in a digestible electronic format, delivered (and consumed) in a fraction of the time!

Ashlock, Paschert, Schuster
Introducing Central Procurement with SAP S/4HANA
https://www.sap-press.com/4800 | $24.99 | 89 pages

Massimo Tuscano
SAP Cloud Platform Mobile Services: Application Development and Operations
https://www.sap-press.com/4781 | $24.99 | 117 pages

Ana Lucia Soler Villanueva
Introducing SAP Jam for SAP SuccessFactors
https://www.sap-press.com/4794 | $24.99 | 126 pages

The Authors of this E-Bite

Wojciech Eichert is an SAP Integration consultant with experience in multiple international projects. He specializes in SAP integration techniques, mainly SAP Application Interface Framework, SAP PI, and SAP PO. Wojciech holds certifications in process integration with SAP NetWeaver 7.31, SAP Fiori implementation and configuration, and ABAP with SAP NetWeaver 7.0.

Krzysztof Łuka is a freelance SAP integration consultant with INT4. He has more than 12 years of experience in IT, 10 of which with SAP. Krzysztof worked on several international implementation projects as developer and integration consultant, making systems integration topics and for the last few years, SAP Application Interface Framework.

Mateusz Nowak is an SAP integration consultant and developer, with experience in several international projects. Mateusz is skilled in a variety of tools and development techniques, especially SAP Application Interface Framework (his blogs on SAP Application Interface Framework topics are available online).

Usage, Service, and Legal Notes

Notes on Usage

This E-Bite is **protected by copyright**. By purchasing this E-Bite, you have agreed to accept and adhere to the copyrights. You are entitled to use this E-Bite for personal purposes. You may print and copy it, too, but also only for personal use. Sharing an electronic or printed copy with others, however, is not permitted, neither as a whole nor in parts. Of course, making them available on the Internet or in a company network is illegal.

For detailed and legally binding usage conditions, please refer to the section Legal Notes.

Service Pages

The following sections contain notes on how you can contact us.

Praise and Criticism

We hope that you enjoyed reading this E-Bite. If it met your expectations, please do recommend it. If you think there is room for improvement, please get in touch with the editor of the book: Will Jobst (willj@rheinwerk-publishing.com).

We welcome every suggestion for improvement but, of course, also any praise! You can also share your reading experience via Twitter, Facebook, or email.

Supplements

If there are supplements available (sample code, exercise materials, lists, and so on), they will be provided in your online library and on the web catalog page for this book. You can directly navigate to this page using the following link: http://www.sap-press.com/4714. Should we learn about typos that alter the meaning or content errors, we will provide a list with corrections there, too.

Technical Issues

If you experience technical issues with your e-book or e-book account at SAP PRESS, please feel free to contact our reader service: support@rheinwerk-publishing.com.

About Us and Our Program

The website http://www.sap-press.com provides detailed and first-hand information on our current publishing program. Here, you can also easily order all of our books and e-books. Information on Rheinwerk Publishing Inc. and additional contact options can also be found at http://www.sap-press.com.

Legal Notes

This section contains the detailed and legally binding usage conditions for this E-Bite.

Copyright Note

Your Rights as a User

You are entitled to use this E-Bite for personal purposes only. In particular, you may print the E-Bite for personal use or copy it as long as you store this copy on a device that is solely and personally used by yourself. You are not entitled to any other usage or exploitation.

In particular, it is not permitted to forward electronic or printed copies to third parties. Furthermore, it is not permitted to distribute the E-Bite on the Internet, in intranets, or in any other way or make it available to third parties. Any public exhibition, other publication, or any reproduction of the E-Bite beyond personal use are expressly prohibited. The aforementioned does not only apply to the E-Bite in its entirety but also to parts thereof (e.g., charts, pictures, tables, sections of text). Copyright notes, brands, and other legal reservations as well as the digital watermark may not be removed from the E-Bite.

Digital Watermark

This E-Bite copy contains a **digital watermark**, a signature that indicates which person may use this copy. If you, dear reader, are not this person, you are violating the copyright. So please refrain from using this E-Bite and inform us about this violation. A brief email to info@rheinwerk-publishing.com is sufficient. Thank you!

Limitation of Liability

Regardless of the care that has been taken in creating texts, figures, and programs, neither the publisher nor the author, editor, or translator assume any legal responsibility or any liability for possible errors and their consequences.

Imprint

This E-Bite is a publication many contributed to, specifically:

Editor Will Jobst
Acquisitions Editor Hareem Shafi
Copyeditor Melinda Rankin
Cover Design Graham Geary
Icon made by Freepik from www.flaticon.com
Layout Design Graham Geary
Production Hannah Lane
Typesetting SatzPro, Krefeld (Germany)

ISBN 978-1-4932-1750-2

1st edition 2019