BIT750

SAP Application Interface Framework

EXERCISES AND SOLUTIONS

Course Version: 03
Course Duration: 17 Hours 55 Minutes

SAP Copyrights, Trademarks and Disclaimers

Typographic Conventions

American English is the standard used in this handbook.

The following typographic conventions are also used.

This information is displayed in the instructor's presentation	
Demonstration	
Procedure	
Warning or Caution	
Hint	
Related or Additional Information	
Facilitated Discussion	
User interface control	*Example text*
Window title	*Example text*

Contents

Unit 1
Exercise 1

Perform Interface Monitoring Without AIF

Business Scenario

To get a good idea of the benefits of AIF, you will perform error handling one time in the traditional way, and one time with AIF. In this exercise you will perform error handling without AIF. The error handling with AIF is performed in the exercise **Manage IDoc Errors**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Perform Error Handling in the Traditional Way

1. Note the time when you start the exercise. Note the time again when the exercise finishes. How many IDocs did you solve?

   Start time: _____

   End time: _____

2. Fix the errors.

   There are the following two possible errors in the generated IDocs:

   - First error: The MATKL Material Group is missing.
   - Second error: The MEINS Base Unit of Measure contains an incorrect entry.

   To fix the error, use the following data:

Field	Value
MATKL	002
MEINS	KG

3. Process the IDoc.

   How long would it take to correct all errors manually?

   _____

   _____

   _____

Unit 1
Solution 1

Perform Interface Monitoring Without AIF

Business Scenario

To get a good idea of the benefits of AIF, you will perform error handling one time in the traditional way, and one time with AIF. In this exercise you will perform error handling without AIF. The error handling with AIF is performed in the exercise **Manage IDoc Errors**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Perform Error Handling in the Traditional Way

1. Note the time when you start the exercise. Note the time again when the exercise finishes. How many IDocs did you solve?

   Start time: _____

   End time: _____

   a) Several IDocs have been created for your user. Display the IDocs by going to transaction `BD87`.

   b) Select your IDocs with the message type **`ZAIF_S###_MATMAS`**.

   c) Select one of the erroneous IDocs and solve the error.

   d) Reprocess the IDoc afterwards.

   e) The node text “Message have been issued: number <log number>” belongs to one IDoc. Double click on one of the IDocs in the tree.

   f) In the next screen, a table displays. Double click on the IDoc number.

   g) On the *IDoc Display* screen, you can see the IDoc’s content and the status records.

2. Fix the errors.

   There are the following two possible errors in the generated IDocs:

   - First error: The MATKL Material Group is missing.
   - Second error: The MEINS Base Unit of Measure contains an incorrect entry.

   To fix the error, use the following data:

Field	Value
MATKL	002

MEINS	KG

a) To figure out which error occurred, expand the Status records node.

b) Double click on node *51* (Application document not posted).

c) On the next screen, click the button *Application Log* in the application toolbar.
You should see the application log that was created. There should be one of the error messages mentioned above.

d) Return to the *IDoc Display* screen.

e) In the data records navigate to segment *E1MARAM* in the selected IDoc.

f) In the Menu Data Record, select *Display -> Change*, switch to the change mode for the segment, and fix the error.

g) If *MATKL* is empty, enter **002**. If *MEINS* contains *K* , change it to **KG**.

h) Save the changes.

3. Process the IDoc.

a) Go back to the IDoc list.

b) Right click on your IDoc and select *Process*.
The IDoc should process successfully.

How long would it take to correct all errors manually?

Endless

Unit 1
Exercise 2

Manage IDoc Errors

Business Scenario

In this exercise, you will work on IDoc errors with AIF.

> Note:
> In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Manage IDoc Errors with AIF

1. Navigate to the Interface Monitor and make yourself familiar with the transaction.
2. Which namespaces and which interfaces do you see in the Interface Monitor?

   _____
   _____
   _____
3. How many messages exist for your interfaces (in total, error, and success)?

   _____
   _____
   _____

> Note:
> Depending on the number of messages the trainer generated, these numbers might differ. Furthermore, the number of errors you fixed in the first exercise will also influence the number of successful and error messages for your MATMAS interface.

4. What is the overall status of your interfaces?
   Answer:_____
5. On which dates have messages been processed?
   Answer:_____

Task 2: Perform Monitoring and Error Handling

1. Find out what type of errors the messages have and correct them if possible. If the same error exists more than once you can use the mass error handling functionality to correct more than one message in one step to save time.

2. Which log messages are displayed?
   Answer:_____

Task 3: Fix a Single Error

1. On the selection screen of Monitoring and Error Handling, select your MATMAS interface and make sure that you select all statuses. Open the messages for your MATMAS interface in Monitoring and Error Handling. Select one of your erroneous IDocs. Fix the error and restart the message.

Task 4: Fix Mass Errors

1. As there are many messages with the wrong material group and many messages with the wrong unit of measurement you don't want to fix all this errors one by one. Use the mass change function and the mass restart function to fix all of them in one step.
2. How long did it this time take to fix the errors?
   Answer:_____

Unit 1
Solution 2

Manage IDoc Errors

Business Scenario

In this exercise, you will work on IDoc errors with AIF.

> Note:
> In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Manage IDoc Errors with AIF

1. Navigate to the Interface Monitor and make yourself familiar with the transaction.
   a) Open the Interface Monitor via transaction `/AIF/IFMON` or start the Interface Monitor via the corresponding entry in the *SAP Menu → Cross-Application Components → SAP Application Interface Framework → Interface Monitor*.
2. Which namespaces and which interfaces do you see in the Interface Monitor?

   There is one namespace displayed which corresponds to your group number S###. The Namespace S### with the long text AIF_S### contains an interface MATMAS/1. Only you are responsible for the interface within your namespace, therefore, only you see it in the Interface Monitor. The other participants have their own namespace with their own interface.
3. How many messages exist for your interfaces (in total, error, and success)?

   The number of messages for the Interface S###/MATMAS/1 depends on the number of messages the trainer generated

> Note:
> Depending on the number of messages the trainer generated, these numbers might differ. Furthermore, the number of errors you fixed in the first exercise will also influence the number of successful and error messages for your MATMAS interface.

4. What is the overall status of your interfaces?
   Answer:_____

a) In the *Status* column, you see the status of your interfaces. If the red light displays, messages with errors exist. If the status icon is green, there are no messages with errors; In this case, all messages have been processed successfully or canceled.

b) At the beginning of this exercise the status is red because there are IDocs with errors.

5. On which dates have messages been processed?

   Answer:_____

   a) On the left hand side, click on the *Without date restriction* button and a calendar control displays.

   b) The current date and/or some days in the past should be highlighted in red (if there are no messages with errors, the date is highlighted in green). On the dates that are not highlighted, no messages have been processed that are within your area of responsibility.

Task 2: Perform Monitoring and Error Handling

1. Find out what type of errors the messages have and correct them if possible. If the same error exists more than once you can use the mass error handling functionality to correct more than one message in one step to save time.

   a) In the Interface Monitor, choose your *MATMAS* interface (namespace **s###**).

   b) Navigate from the Interface Monitor to Monitoring and Error Handling by double clicking on the *Sum* sign at the right side of the screen.

   Monitoring and Error Handling opens and displays the selected messages.

2. Which log messages are displayed?

   Answer:_____

   a) Double click on the interface *MATMAS*.

   b) The messages that occurred during processing of the IDocs should be loaded into application log view. For example, the following messages could be displayed:

   - The field MARA-MATKL is defined as a required field; it does not contain an entry.
   - Unit of measure K is not defined as a commercial unit.
   - Trying to change: ZS###_MAT_9.
   - Messages have been issued: number &.

   If there have been errors, per default, only the error messages are displayed.

   c) Click on one message in the *Application Log*.

   The corresponding message should be displayed in the Log Messages view at the left bottom of the screen.

   d) Double click on some structure or tables in the Data Structure view.

   The corresponding content displays in the *Data Content* view in the right bottom of the screen. If you have several messages selected, the data of all selected messages displays.

   e) To see only errors return to the previous screen. This time double click on the red light at the message screen . In the *Data Content* view, you should only see messages that have an error status.

Task 3: Fix a Single Error

1. On the selection screen of Monitoring and Error Handling, select your MATMAS interface and make sure that you select all statuses. Open the messages for your MATMAS interface in Monitoring and Error Handling. Select one of your erroneous IDocs. Fix the error and restart the message.
   a) In the *Monitoring and Error Handling* screen, select one of your erroneous IDocs.
   b) Double click on the first erroneous IDoc.
   c) Check which error occurred in the *Log Messages* view.
   d) Double click on the *E1MARAM* node in the Data Structure view.
      The content of the segment displays in the *Data Content* view.
   e) Search for the erroneous field and change the content (if unit of measure is wrong, enter **KG** in the *MEINS* field ; If the Material Group is missing, enter 002 in field MATKL). To change the data, double click on the corresponding cell.
   f) Enter the correct value in the popup.
   g) Save the changes.
      The icon in the *Data Messages* view indicates that the message was changed.
   h) In the *Data Messages* view, click *Restart*.
      The message will process successfully.

Task 4: Fix Mass Errors

1. As there are many messages with the wrong material group and many messages with the wrong unit of measurement you don't want to fix all this errors one by one. Use the mass change function and the mass restart function to fix all of them in one step.
   a) Select all messages by clicking on the interface name instead of a specific message in the *Data Messages* view.
   b) Double click on *E1MARAM* in the *Data Structure* view.
   c) Select the *MEINS* column and press the *Search & Replace* button.
   d) Enter **K** in the *Find* field and enter '**KG**' in the *Replace* field.
   e) Press *Check*. The number of the fields that will be changed display. Click *OK*.
   f) If you deselect the column, you see that the cells that have been changed are highlighted in yellow.
   g) Save the changes.
   h) The messages that have been changed get the corresponding icon.
   i) Restart all messages.
   j) If you already had successful messages in your list and tried to restart those messages, you get an error message with the information that successfully executed messages could not be reprocessed. Ignore this message.
2. How long did it this time take to fix the errors?

Answer:_____

a) Note down the required time into the line above.

Unit 2
Exercise 3

Create an Interface

Business Scenario

in this exercise, you will create a first interface.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create Structures

1. You need one RAW Structure and one SAP Structure for your interface. Create the following structures by copying the example structures ZAIF_T100_RAW_FLIGHT and ZAIF_T100_SAP_FLIGHT. In both change T100 to S### in the target structure and activate the structures.

Task 2: Check a Namespace

1. Namespace S### was already created for you, therefore, you do not have to create a new one. So you just have to check the namespace.

Task 3: Create an Interface

1. Create your interface FLBOOKING with the version 1 in the AIF Customizing menu /AIF/CUST. Use the Structures you have created as the RAW and the SAP Structure

   Use the following data:

Field	Value
Interface Name	FLBOOKING
Interface Version	1
Description	<any>
SAP Data Structure	ZAIF_S###_SAP_FLIGHT
Raw Data Structure	ZAIF_S###_RAW_FLIGHT

Task 4: Set Engines

1. For the example interface created during this training, we use AIF's own XML runtime and persistence. Therefore, we need an interface using the XML engines.

Task 5: Create Recipients for All Interfaces

1. We require a recipient to be able to display the messages processed during this training in the Interface Monitor. Name it ALL_INTERFACES and assign your user to it. It shall be an Receiver for all interfaces within your namespace.

   Use the following data:

Field	Value
Recipient for Alert	ALL_INTERFACES
Alert Recipient Description	<any>, e.g. recipient for all interfaces

2. Assign the recipient.
3. Assign further recipients.

   Use the following data:

Field	Value
User Name	<Your_User> (AIF_S###)
Message Type	None
Include in Overview Screen	Select
Technical User	Select

Unit 2
Solution 3

Create an Interface

Business Scenario

in this exercise, you will create a first interface.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create Structures

1. You need one RAW Structure and one SAP Structure for your interface. Create the following structures by copying the example structures ZAIF_T100_RAW_FLIGHT and ZAIF_T100_SAP_FLIGHT. In both change T100 to S### in the target structure and activate the structures.
   a) Start transaction `SE11`.
   b) In the *Data Type* field fill in the name of the target structure **`ZAIF_T100_RAW_FLIGHT`**.
   c) Click the *Copy* button.
   d) Change the name in the *Structure* field to **`ZAIF_S###_RAW_FLIGHT`**.
   e) Click *Continue*.
   f) On the following popup add your package *ZAIF_S###* and click *Save*.
   g) In the next popup enter a development request. The F4-Help shows you the ones you can use. Choose it and press *Enter*.
      The System informs, that the structure was copied successfully.
   h) Press *Activate* or *CTRL+F3* to finish your structure.
   i) Perform the steps a to h for the second structure.

Task 2: Check a Namespace

1. Namespace S### was already created for you, therefore, you do not have to create a new one. So you just have to check the namespace.
   a) Go to AIF Customizing (transaction `/AIF/CUST`). Under *Interface Development* → *Define Namespace* check if your namespace is existing.
      Here you find the long text of the namespace that you have seen in the Interface Monitor.

Note:
If you want to navigate to the AIF Customizing via transaction /AIF/CUST, you have to add /n or /o in front of /AIF/CUST.

Task 3: Create an Interface

1. Create your interface FLBOOKING with the version 1 in the AIF Customizing menu /AIF/CUST. Use the Structures you have created as the RAW and the SAP Structure

   Use the following data:

Field	Value
Interface Name	FLBOOKING
Interface Version	1
Description	<any>
SAP Data Structure	ZAIF_S###_SAP_FLIGHT
Raw Data Structure	ZAIF_S###_RAW_FLIGHT

   a) Go to AIF Customizing (transaction */AIF/CUST*).

   b) Navigate to *Interface Development → Define Interfaces*.

   c) Enter your namespace. If there is only one interface existing in this namespace it is directly shown. Don't change something in here.

   d) Select the *New Entries* button.

   e) Into the pop-up, enter the data from the table above.

   f) Save.

Task 4: Set Engines

1. For the example interface created during this training, we use AIF's own XML runtime and persistence. Therefore, we need an interface using the XML engines.

   a) Go to AIF Customizing (transaction `/AIF/CUST`).

   b) Choose *Interface Development → Additional Interface Properties → Specify Interface Engines*.

   c) Enter your namespace.

   d) Change the value for *APPLICATION ENGINE* and for *PERSISTENCE ENGINE* from *Proxy* to **`XML`**.

   e) Save.

Task 5: Create Recipients for All Interfaces

1. We require a recipient to be able to display the messages processed during this training in the Interface Monitor. Name it ALL_INTERFACES and assign your user to it. It shall be an Receiver for all interfaces within your namespace.

   Use the following data:

Field	Value
Recipient for Alert	ALL_INTERFACES
Alert Recipient Description	<any>, e.g. recipient for all interfaces

   a) Go to AIF Customizing (transaction `/AIF/CUST`).

   b) Navigate to *Error Handling* → *Namespace-Specific Features*.

   c) Enter your namespace (**s###**).

   d) Select *Define Recipients*

   e) Create a new entry and use the data from the table above.

   f) Save.

2. Assign the recipient.

   a) Go to AIF Customizing (transaction `/AIF/CUST`).

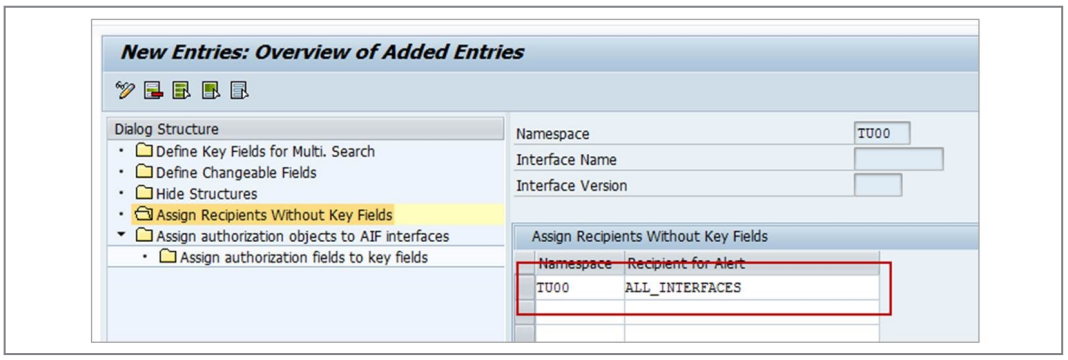
   b) Navigate to *Error Handling* → *Interface-Specific Features*.

   c) Enter your namespace. Leave the interface name and version fields blank to assign it to all the interfaces in this namespace.

   d) Go to *Assign Recipients without Key Fields* and create a new entry.

   e) Enter your namespace and the value you defined above for *Recipient for Alert* (you can select the entry using value help).

   f) Save.



3. Assign further recipients.

   Use the following data:

Field	Value
User Name	<Your_User> (AIF_S###)
Message Type	None
Include in Overview Screen	Select

Field	Value
Technical User	Select

a) Go to AIF Customizing (transaction `/AIF/CUST`).

b) Navigate to *System Configuration* → *Assign Recipients*.

c) Enter your namespace and the recipient.

d) In *Assign Users* create a new entry and enter the data from the table above.

e) Save.

Unit 3
Exercise 4

Create and Test Mappings for a Sales Agent Structure

Business Scenario

In this exercise, you will use simple Mapping functionality with Move-Corresponding and Move.

> Note:
> In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

> Note:
> To perform this exercise, the exercise **Create Basic AIF Elements** must be finished successfully, and there must be no copy template.

Task 1: Create a Structure Mapping for the SALES_AGENT Structure

1. You want to move the SALES_AGENT-Structure from the source Interface to the SALES_AGENT-Structure from the target Interface. The Structures have the same definition which makes it possible to move them all in once.

Task 2: Test Your Mapping

1. Use the AIF Test Tool to check your mapping.
   Use the following data:

Field	Value
SALES_AGENT_ID	1
NAME	Harry Urlaub
PHONE	123456
EMAIL	harry@urlaub.de

Task 3: Create a Mapping for CUSTOMER_DATA Structure

1. In the next step, you want to map the customer structures. They have a different definition, so you have to maintain a mapping for every needed field. Do a simple 1:1 mapping of the customer fields. The Source Structure is CUSTOMER_DATA, the target Structure is CUSTOMERS.

Create the following fields:

Field in Destination Structure field	Field Name 1 field (in the Define Field Mappings area)
CUSTOMER_DATA-CITY	CUSTOMER_DETAILS-CITY
CUSTOMER_DATA-COUNTR	CUSTOMER_DETAILS-COUNTRY
CUSTOMER_DATA-CUSTNAME	CUSTOMER_DETAILS-NAME
CUSTOMER_DATA-CUSTTYPE	CUSTOMER_DETAILS-CUSTTYPE
CUSTOMER_DATA-EMAIL	CUSTOMER_DETAILS-EMAIL
CUSTOMER_DATA-PHONE	CUSTOMER_DETAILS-TELEPHONE
CUSTOMER_DATA-POSTCODE	CUSTOMER_DETAILS-POSTCODE
CUSTOMER_DATA-STREET	CUSTOMER_DETAILS-STREET
CREDITCARD	CUSTOMER_DETAILS-CREDITCARD

Task 4: Test Your Mapping

1. Test the Mapping of the Customer Structures in the AIF Interface Test tool.
   Use the following data:

Name	Street	Post Code	City	Country	Phone	Custtype	Email	Class
Agathe Gewitter	Kaiserstr. 140	76133	Karlsruhe	DE	072156834	Standard	agathe@gewitter.de	C
Achim Müller	Durlacher Str. 5	76646	Bruchsal	DE	07251908056	Company	Mueller@gmbh.de	F

   Also enter a value in the credit card field and save you new testfile.

Task 5: Use the Different Ways to Test

You always have the following two options to test your interface:

- Option 1: Use the AIF interface test tool to create your test data and check the results.
- Option 2: Use the AIF interface test tool to create your test data and use the XML runtime and the Interface Monitor to execute and check the data.

1. Process File via Interface Test Tool.
2. Process your test file via the XML runtime and monitor the errors in the Interface Monitor.

Unit 3
Solution 4

Create and Test Mappings for a Sales Agent Structure

Business Scenario

In this exercise, you will use simple Mapping functionality with Move-Corresponding and Move.

> Note:
> In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

> Note:
> To perform this exercise, the exercise **Create Basic AIF Elements** must be finished successfully, and there must be no copy template.

Task 1: Create a Structure Mapping for the SALES_AGENT Structure

1. You want to move the SALES_AGENT-Structure from the source Interface to the SALES_AGENT-Structure from the target Interface. The Structures have the same definition which makes it possible to move them all in once.
   a) In AIF Customizing (Transaction `/AIF/CUST`), go to *Interface Development* → *Define Structure Mappings*.
   b) Enter your namespace (*S###*), the name of the interface (*FLBOOKING*) and the interface version (1) created before.
   c) Press *New Entry* to create a new source structure.
   d) In the table on the righthand side, select the first empty row and press *F4*. Select the *SALES_AGENT* structure. Press *Enter*.
   e) Select the newly created *SALES_AGENT* entry and click on *Assign Destination Structures* on the left menu path.
   f) Map the SALES_AGENT of the raw structure to the SALES_AGENT structure of the SAP data structure. Click NEW ENTRIES. Enter **`10`** as the number of Structure Mapping and select the SALES_AGENT of the SAP data structure using the value help (F4) of the destination structure field.
   g) Select the *Move Corresponding Fields* checkbox on this screen.
   As the SALES_AGENT source structure and the destination structure are identical, you can use the *Move Corresponding Fields* functionality to map the fields of the structure.

h) Save.

Task 2: Test Your Mapping

1. Use the AIF Test Tool to check your mapping.
   Use the following data:

Field	Value
SALES_AGENT_ID	1
NAME	Harry Urlaub
PHONE	123456
EMAIL	harry@urlaub.de

   a) Go to AIF interface test tool (transaction `/AIF/IFTEST`).

   b) On the *Select Test Files* screen, press *Search*.

   c) Select *New Data* and choose your namespace and interface created before.

   d) Enter a description.

   e) Choose *Save*.

   f) Select the created file and press *READ DATA*.

   g) If you receive the warning **`Inbound or Outbound must be maintained`** ignore it by pressing *Enter*.

   h) In the raw structure view, double click on the *SALES_AGENT* structure. Fill in the above entries for the fields you want to check and press *Save*.

   i) Press *TRANSFORM*.

   j) Check the result in the SAP structure (displayed in the second structure of the screen).
      The sales agent information should be available in the SAP structure.

Task 3: Create a Mapping for CUSTOMER_DATA Structure

1. In the next step, you want to map the customer structures. They have a different definition, so you have to maintain a mapping for every needed field. Do a simple 1:1 mapping of the customer fields. The Source Structure is CUSTOMER_DATA, the target Structure is CUSTOMERS.
   Create the following fields:

Field in Destination Structure field	Field Name 1 field (in the Define Field Mappings area)
CUSTOMER_DATA-CITY	CUSTOMER_DETAILS-CITY
CUSTOMER_DATA-COUNTR	CUSTOMER_DETAILS-COUNTRY
CUSTOMER_DATA-CUSTNAME	CUSTOMER_DETAILS-NAME

Field in Destination Structure field	Field Name 1 field (in the Define Field Mappings area)
CUSTOMER_DATA-CUSTTYPE	CUSTOMER_DETAILS-CUSTTYPE
CUSTOMER_DATA-EMAIL	CUSTOMER_DETAILS-EMAIL
CUSTOMER_DATA-PHONE	CUSTOMER_DETAILS-TELEPHONE
CUSTOMER_DATA-POSTCODE	CUSTOMER_DETAILS-POSTCODE
CUSTOMER_DATA-STREET	CUSTOMER_DETAILS-STREET
CREDITCARD	CUSTOMER_DETAILS-CREDITCARD

a) Go to AIF Customizing (transaction `/AIF/CUST`).

b) Go to *Interface Development → Define Structure Mappings*.

c) Select your namespace and interface created before.

d) Create a new entry. Using value help, select **`CUSTOMER_DATA`** as *Source Structure*.

e) Press *Enter*.

f) Select the *CUSTOMER_DATA* entry

g) Select *Assign Destination Structure* on the left.

h) Create a new entry. Enter 10 as number of structure mapping and select **`CUSTOMERS`** as *destination structure* (the value help of the destination structure field can be used to select the correct structure).

i) Select *Define Field Mappings* on the left hand side to create the field mappings listed above.

j) Click on *NEW ENTRY* and create the first field mapping. If you want to create the next field mappings navigate to an empty field mapping using the *Next Entry* button.

k) When you are finished with all entries press *Save*.

l) If you are asked for a package user *ZAIF_S###*, if you are asked for a transport use the F4 Help to find your transport request.

Task 4: Test Your Mapping

1. Test the Mapping of the Customer Structures in the AIF Interface Test tool.

   Use the following data:

Name	Street	Post Code	City	Country	Phone	Custtype	Email	Class
Agathe Gewitter	Kaiserstr. 140	76133	Karlsruhe	DE	072156834	Standard	agathe@gewitter.de	C

Name	Street	Post Code	City	Country	Phone	Custtype	Email	Class
Achim Müller	Durlacher Str. 5	76646	Bruchsal	DE	07251908056	Company	Mueller@gmbh.de	F

Also enter a value in the credit card field and save you new testfile.

a) Go to the AIF interface test tool (`/AIF/IFTEST`).

You will find your file in the list at the bottom of the screen.

b) Double click on the file and press the Read Data *Read Data button*.

c) Confirm the displayed warning.

d) Extend the previously created test file and add customer information in the raw structure.

e) Double click on *CUSTOMER_DATA* and press *ADD*.

f) Double click on *CUSTOMER_DETAILS* and enter the details displayed above.

g) Repeat the steps for the second customer. After you have entered the details for the first customer, you have to navigate back from the *CUSTOMER_DETAILS* in the raw data structure to the *CUSTOMER_DATA* structure via the *Back* button.

Task 5: Use the Different Ways to Test

You always have the following two options to test your interface:

- Option 1: Use the AIF interface test tool to create your test data and check the results.
- Option 2: Use the AIF interface test tool to create your test data and use the XML runtime and the Interface Monitor to execute and check the data.

1. Process File via Interface Test Tool.

a) Transform and process the file via the AIF interface test tool (transaction `/AIF/IFTEST`).

b) Check the results in the SAP structure part of the interface test tool.

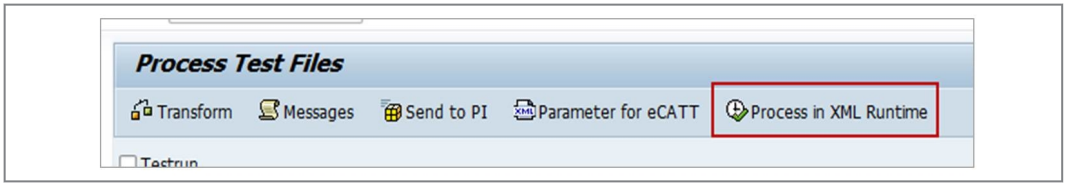
In the SAP structure, *CUSTOMER_DATA* should contain the two customer entries created above.

2. Process your test file via the XML runtime and monitor the errors in the Interface Monitor.

a) Open the test file via the AIF interface test tool (transaction `/AIF/IFTEST`).

b) To monitor the interface in the Interface Monitor, process the file via the *Process in XML Runtime* button.

This button creates an XML message from the file and uses the AIF XML engine to process the data.



c) Open the Interface Monitor via transaction */AIF/IFMON*.

d) Select the icon  .

You can see the created test messages in the *Data Messages* view. All messages found for the selected interface are displayed. If you have selected more than one interface, the messages of the different interfaces are displayed under the corresponding interface.

e) Select one of your messages.

f) In the *Data Structure* view on the bottom left side of the screen, you can see your data structure.

The raw structure displays.

g) If you click on one of the structures, the content displays in the *Data Content* view on the right side of the *Data Structure* view.

h) You can switch between displaying the destination and the source structure. If you want to display the destination structure, you have to switch to *Technical Mode*.

i) You have 2 buttons - *Transform* and *Switch* – above the *Data Structure* (if the *Switch* button is not displayed, click on *Transform*).

j) Click *Switch*.

The SAP structure displays.

k) Click on *CUSTOMER_DATA* and check the result of your mapping.

Unit 3
Exercise 5

Create Checks and Fix Value Mappings

Business Scenario

In this exercise, you will create several checks and you will fix values, which are detected by these checks.

Note:
This exercise depends on completing the exercise **Create and Test Mapping for a Sales Agent Structure**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create Field Check for Sales Agent

1. Create a check.

   Before we process the sales agent data, we ensure that the sales agent does not already exist in the system. If the sales agent is already known, skip the processing. A sales agent exists if *SALES_AGENT_ID* contains an entry.

   Use the following data:

Field	Value
Number of check	**10**
Check Raw Data	Select
Ignore Data If Check Is Not Successful	**Ignore data if check is not successful (no error)**
Field Name 1	**SALES_AGENT_ID**
Field Check	**Empty**

2. Test your development.

   Use the AIF Test Tool (Transaction `/AIF/IFTEST`).

   Use the following data:

Field	Value
Sales agent ID	1

Task 2: Validate Travel Agency Existence via Check

You want to validate the existence of the travel agency. Determine the correct agency and check if the agency exists. Also, you want to use user-specific error messages.

1. Assign a check.

   Use the following data:

Field	Value
Source Structure	ZAIF_S###_RAW_FLIGHT
Destination Structure	
Number of Structure Mapping	**10**
Destination Structure	ZAIF_S###_SAP_FLIGHT
Assign Checks	
Number of Check	**10**
Namespace	Your namespace
Name for the check	**AGENCY_EXISTS**
Agency	
Ignore Data If Check Is Not Successful	**Treat as error if check is not successful**
Check Raw data	Select
Field Name 1	**TRAVEL_AGENCY-AGENCYNUM**

2. Create a check with a user-specific error message.

   Display the error messages in the case of an error with the travel agency.

   Use the following data:

Field	Value
Message class	ZAIF_TRAINING
Entry	Existing 001
Error	
Error Msg. Class	**ZAIF_TRAINING**
Error Msg. Number	**001**
Variable Definition [first]	**$1**
Table Check	
Number of check	**10**
Table Name	**STRAVELAG**
Where Condition	**AGENCYNUM = '$1'**
Check Existence in Table	Select

3. Test your mapping and the newly created check.
   Use the following data:

Field	Value
Travel agency ID	101

Task 3: Create a Fix Value Mapping

1. Create a fix value for the customer discount.
   Use the following data:

Field	Value
Customer discount	10
Fix value name	CUSTOMER_DISCOUNT
Source Structure	CUSTOMER_DATA
Discount	
Field Name	**CUSTOMER_DATA-DISCOUNT**
Namespace	Your namespace
Field Name of Fix Value	**CUSTOMER_DISCOUNT**
Value	10

2. Test the Fix ValueMapping.
   Use the AIF test tool to check your fixed value.
3. Optional step: Process the data using Process in XML Runtime.
   To see the created message, open the Interface Monitor via transaction `/AIF/IFMON`.

Unit 3
Solution 5

Create Checks and Fix Value Mappings

Business Scenario

In this exercise, you will create several checks and you will fix values, which are detected by these checks.

> Note:
> This exercise depends on completing the exercise **Create and Test Mapping for a Sales Agent Structure**.

> Note:
> In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create Field Check for Sales Agent

1. Create a check.

   Before we process the sales agent data, we ensure that the sales agent does not already exist in the system. If the sales agent is already known, skip the processing. A sales agent exists if *SALES_AGENT_ID* contains an entry.

   Use the following data:

Field	Value
Number of check	**10**
Check Raw Data	Select
Ignore Data If Check Is Not Successful	**Ignore data if check is not successful (no error)**
Field Name 1	**SALES_AGENT_ID**
Field Check	**Empty**

   a) Navigate to *Interface Development → Define Structure Mapping → AIF Customizing (transaction /AIF/CUST)*.

   b) Select the *Sales Agent* structure and choose *Assign Checks* on the left.

   c) To create a new entry, enter and select the values provided in the table.

   d) As detailed in the table, select the *Check Raw Data* checkbox.

Check Raw Data executes the check before the mapping. The *Ignore Data If Check Is Not Successful* selection ensures that the sales agent data is not processed if the sales agent already exists (in case the sales agent ID contains an entry).

e) Choose *Save*.

2. Test your development.

Use the AIF Test Tool (Transaction `/AIF/IFTEST`).

Use the following data:

Field	Value
Sales agent ID	1

a) Navigate to the AIF interface test tool (Transaction `/AIF/IFTEST`).

b) Search for your file.

c) Double click on the file and choose *Read Data* 

.

d) Confirm the displayed warning.

e) Check the entry in the sales agent structure. If the sales agent ID is empty, enter the value provided in the step.

f) Choose *Transform* and check the result.
The mapped sales agent structure is empty.

g) Optional step: You can process the test file, as described in the previous exercise. Check the result in Monitoring and Error Handling.
The sales agent structure is empty.

h) Return to the sales agent structure in your test file and remove the sales agent ID.

i) Transform the file again.
The sales agent structure is mapped.

Task 2: Validate Travel Agency Existence via Check

You want to validate the existence of the travel agency. Determine the correct agency and check if the agency exists. Also, you want to use user-specific error messages.

1. Assign a check.

Use the following data:

Field	Value
Source Structure	ZAIF_S###_RAW_FLIGHT
Destination Structure	
Number of Structure Mapping	`10`
Destination Structure	ZAIF_S###_SAP_FLIGHT
Assign Checks	
Number of Check	`10`
Namespace	Your namespace

Field	Value
Name for the check	**AGENCY_EXISTS**
Agency	
Ignore Data If Check Is Not Successful	**Treat as error if check is not successful**
Check Raw data	Select
Field Name 1	**TRAVEL_AGENCY-AGENCYNUM**

a) Navigate to *Interface Development → Define Structure Mapping → AIF Customizing (transaction /AIF/CUST)*.

b) To create a new entry for source structures, select the root structure.

c) For the *Source Structure* field, enter the data provided in the step and choose *Enter*.

d) Select the new entry and go to *Assign Destination Structure*.

e) Enter the data provided in the Destination Structure table and choose *Enter*.

f) Navigate to *Assign Checks*.

g) To create a new entry, enter the data provided in the Assign Checks table and choose *Enter*.

h) Confirm that you want to create the check.

i) Enter the data provided in the Agency table.

If the agency does not exist, this ensures that no processing or mapping is executed. The agency must be checked at the beginning of the processing. You can also use value help to select the correct entry.

j) Choose *Save*.

k) Confirm that you want to create the check.

2. Create a check with a user-specific error message.

Display the error messages in the case of an error with the travel agency.

Use the following data:

Field	Value
Message class	ZAIF_TRAINING
Entry	Existing 001
Error	
Error Msg. Class	**ZAIF_TRAINING**
Error Msg. Number	**001**
Variable Definition [first]	**$1**
Table Check	
Number of check	**10**

Field	Value
Table Name	**STRAVELAG**
Where Condition	**AGENCYNUM = '$1'**
Check Existence in Table	Select

a) Double click the name of the check that was just created. A new window opens.

b) Go to change mode and enter a description for your check.

c) Enter the data provided in the Error table.

d) Choose *Save*.

e) Navigate to *Define Single Check* and create a new entry.

f) Enter the data provided in the Table Check table.

Note:
$1 is a placeholder for the travel agency number, which transfers to the check as defined in the structure mapping view. For **Check Existence in Table**, if there is no travel agency with the given number, the check fails and further processing stops. The previously defined error message displays in the error handling.

g) Choose *Save*.

3. Test your mapping and the newly created check.
Use the following data:

Field	Value
Travel agency ID	101

a) Navigate to AIF interface test tool (transaction `/AIF/IFTEST`) and load your test file.

b) Change the TRAVEL_AGENCY structure to the value provided in the step.

c) Save and transform the file.

d) Process the file in the XML runtime.
The travel agency does not exist and the defined error message displays.

e) Change the travel agency ID to the value provided in the step.

f) Retry the test.
The file processes successfully.

g) Process the data using Process in XML Runtime.

h) Open the Interface Monitor via transaction `/AIF/IFMON`.

i) Select the icon  .
You are forwarded to the Monitoring and Error Handling view.

j) Check the results.

Task 3: Create a Fix Value Mapping

1. Create a fix value for the customer discount.
   Use the following data:

Field	Value
Customer discount	10
Fix value name	CUSTOMER_DISCOUNT
Source Structure	CUSTOMER_DATA
Discount	
Field Name	**CUSTOMER_DATA-DISCOUNT**
Namespace	Your namespace
Field Name of Fix Value	**CUSTOMER_DISCOUNT**
Value	10

   a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Structure Mappings*.

   b) Select your namespace and interface.

   c) Select the Source Structure value provided in the step.

   d) Navigate to *Define Fix Value* and create a new entry.

   e) Enter the data from the Discount table.

   f) Choose *Enter*.

   g) Confirm that you want to create the fix value.

   h) Choose *Save*.

   i) Double click on the created fix value.
   The definition of the fix value opens in a new window.

   j) Switch to change mode and enter a description.

   k) Enter the value provided in the step.

   l) Save and close the window.

2. Test the Fix ValueMapping.
   Use the AIF test tool to check your fixed value.
   a) Navigate to the AIF interface test tool transaction (transaction `/AIF/IFTEST`).

   b) Select and transform your file using XML Runtime.

   c) Check the result in the SAP structure.
   In the CUSTOMER_DATA structure, value 10 is listed in the DISCOUNT field.

3. Optional step: Process the data using Process in XML Runtime.
   To see the created message, open the Interface Monitor via transaction `/AIF/IFMON`.

a) Select the icon  .
You are forwarded to the Monitoring and Error Handling view.

b) Check the data.

c) Select the newest message.

d) Activate technical mode and choose *Transform* to display the destination structure.

e) Check if the discount is correctly transferred to the *CUSTOMER_DATA* correctly.

Unit 4
Exercise 6

Create Indirect Mappings and Value Conversion

Business Scenario

We want to map a customer's flight bookings. Also, we want to convert the incoming flight date into the format used in our system via value conversion.

> Note:
> In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

> Note:
> This exercise depends on completing the exercise **Create Checks and Fix Values**.

Task 1: Map Indirectly

To map the customers to the corresponding flight bookings, we need indirect mapping. We create this indirect mapping for the FLIGHT_BOOKING structure and add the necessary corresponding mappings of the higher level structure CUSTOMER_DATA. Before mapping the FLIGHT_BOOKING structures, we must add the corresponding mapping in CUSTOMER_DATA.

1. Perform necessary mapping in CUSTOMER_DATA.

   Use the following data:

Field	Sub-Table
FLIGHT_BOOKINGS	**`FLIGHT_BOOKINGS`**

2. Map the Flight Bookings

   Use the following data:

Field	Value
Number of Structure Mapping	**`10`**
Destination Structure	**`FLIGHT_BOOKINGS`**
Indirect Mapping	Select

3. Create a field mapping.

   Use the following data:

Field	Value
BOOKING_DATA-CLASS	**CLASS**
BOOKING_DATA-AGENCYNUM	**TRAVEL_AGENCY-AGENCYNUM**
LINE_ID	CUSTOMER_DATA!<LINENR>

4. Test the mapping.

   Use the AIF test tool. Use the following data:

Name	*FLDATE*	*CLASS*	*AIRPORT_FROM*	*AIRPORT_TO*
Agathe Gewitter	**09.11.2013**	**Y**	**JFK**	**SFO**
	04.11.2013		**JFK**	**FRA**
Achim Muller	**05.11.2013**		**SIN**	**HKG**

Task 2: Convert Value for the Flight Date

The component type of the FLDATE field is different in the source and destination structure. To prevent errors, convert the incoming date to the data type of the SAP structure.

1. Map the field FLIGHTDATE.

   Use the following data:

Field	Value
Destination Structure	
Field in Destination Structure	**BOOKING_DATA-FLIGHTDATE**
Fieldname 1	**FLDATE**
Define Field Mappings	
Data Element for Conversion	**DATS**
Direction of Conversion Exit	**External ->Internal**

2. Test the value conversion.

Unit 4
Solution 6

Create Indirect Mappings and Value Conversion

Business Scenario

We want to map a customer's flight bookings. Also, we want to convert the incoming flight date into the format used in our system via value conversion.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Note:
This exercise depends on completing the exercise **Create Checks and Fix Values**.

Task 1: Map Indirectly

To map the customers to the corresponding flight bookings, we need indirect mapping. We create this indirect mapping for the FLIGHT_BOOKING structure and add the necessary corresponding mappings of the higher level structure CUSTOMER_DATA. Before mapping the FLIGHT_BOOKING structures, we must add the corresponding mapping in CUSTOMER_DATA.

1. Perform necessary mapping in CUSTOMER_DATA.

   Use the following data:

Field	Sub-Table
FLIGHT_BOOKINGS	**`FLIGHT_BOOKINGS`**

   a) Navigate to *Interface Development → Define Structure Mappings → AIF Customizing (transaction /AIF/CUST)*.

   b) Select your namespace and interface.

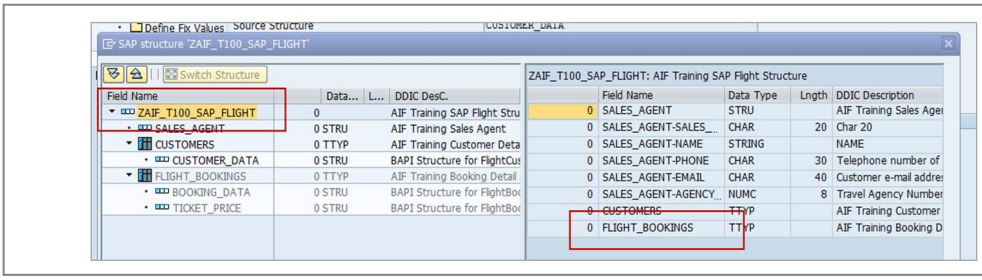
   c) Choose *CUSTOMER_DATA* as source structure.

   d) Choose *Assign Destination Structures* on the left hand side.

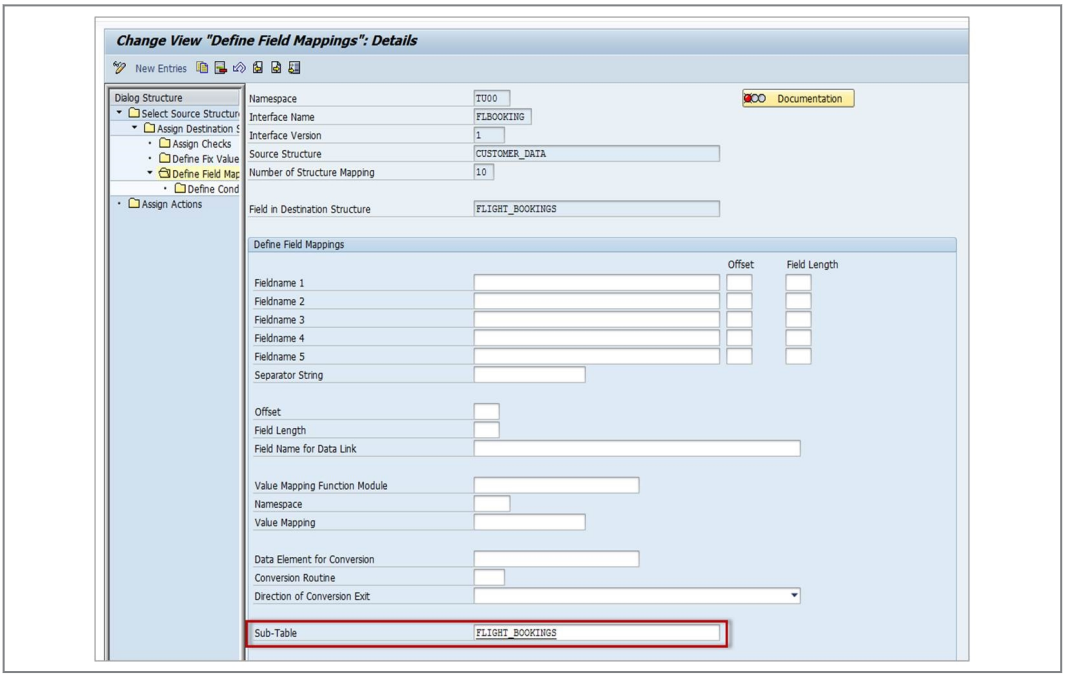
   e) Choose *Define Field Mappings* on the left hand side.

   f) To create a new entry, enter the data from the Destination Structure table.

> Note:
> If you use *F4* help to select the *Field in Destination Structure*, select the root structure on the left hand side of the module window and choose *FLIGHT_BOOKINGS* on the right hand side.



> Note:
> Make sure to enter FLIGHT_BOOKINGS in the Sub-Table field. *Field Name 1* remains empty.



g) Choose *Save*.

2. Map the Flight Bookings

Use the following data:

Field	Value
Number of Structure Mapping	**10**
Destination Structure	**FLIGHT_BOOKINGS**
Indirect Mapping	Select

a) On the left hand side, navigate to *Select Source Structure* and create a new entry for *FLIGHT_BOOKINGS*.

b) Choose *Enter*.

c) Select the created FLIGHT_BOOKINGS entry and navigate to *Assign Destination Structures*.

d) Create a new entry using the data from the step.

3. Create a field mapping.

   Use the following data:

Field	Value
BOOKING_DATA-CLASS	**CLASS**
BOOKING_DATA-AGENCYNUM	**TRAVEL_AGENCY-AGENCYNUM**
LINE_ID	CUSTOMER_DATA!<LINENR>

a) Navigate to *Define Field Mappings* on the left.

b) Using the data from the Flight Bookings table, create the necessary mappings for the FLIGHT_BOOKINGS structure.

c) To map the flights to the corresponding customer, map the line number.

d) Create a new field mapping.

e) Enter the data provided in the table, add the line number of the current *CUSTOMER_DATA* line.
You can add the customer numbers to the corresponding flight bookings. The `<LINENR>` parameter enables you to use the current line number. The exclamation mark *!* enables you to access data from a table in the hierarchy above your current structure. In combination, you can access the current line number of the current CUSTOMER_DATA line.

Note:
F4 help is not available for this mapping.

f) Choose *Save*.

4. Test the mapping.

   Use the AIF test tool. Use the following data:

Name	*FLDATE*	*CLASS*	*AIRPORT_FROM*	*AIRPORT_TO*
Agathe Gewitter	**09.11.2013**	**Y**	**JFK**	**SFO**
	04.11.2013		**JFK**	**FRA**
Achim Muller	**05.11.2013**		**SIN**	**HKG**

a) Navigate to the AIF interface test tool (transaction `/AIF/IFTEST`) and check if the line number is mapped correctly.

b) Add the data for flight bookings from the Customers table.

c) Save the changes.

> Note:
> Possibly, the flight data described above does not work, for example, if the flight is completely occupied. In this case, go to the *SFLIGHTS* table, select flights where seats are still available, and change your test data accordingly.

d) Process the data through the AIF interface test tool.

e) Check entries in the SAP structure.
FLIGHT_BOOKINGS contains 3 entries:

- For customer 1, two entries with LINE_ID 1.
- For customer 2, one entry with LINE_ID 2.

f) Process the data using Process in XML Runtime.

g) Open the Interface Monitor (transaction `/AIF/IFMON`) and select the icon  .
You are forwarded to the Monitoring and Error Handling View.

h) Check the status of the message.
The message is processed successful.

i) Check the result of the mapping in the Data Structure and Data Content views.

Task 2: Convert Value for the Flight Date

The component type of the FLDATE field is different in the source and destination structure. To prevent errors, convert the incoming date to the data type of the SAP structure.

1. Map the field FLIGHTDATE.
   Use the following data:

Field	Value
Destination Structure	
Field in Destination Structure	**BOOKING_DATA-FLIGHTDATE**
Fieldname 1	**FLDATE**
Define Field Mappings	
Data Element for Conversion	**DATS**
Direction of Conversion Exit	**External ->Internal**

a) Navigate to *Interface Development → Define Structure Mappings → AIF Customizing (transaction* `/AIF/CUST`*)*.

b) Enter your namespace and interface.

c) Select *FLIGHT_BOOKING* as the source structure.

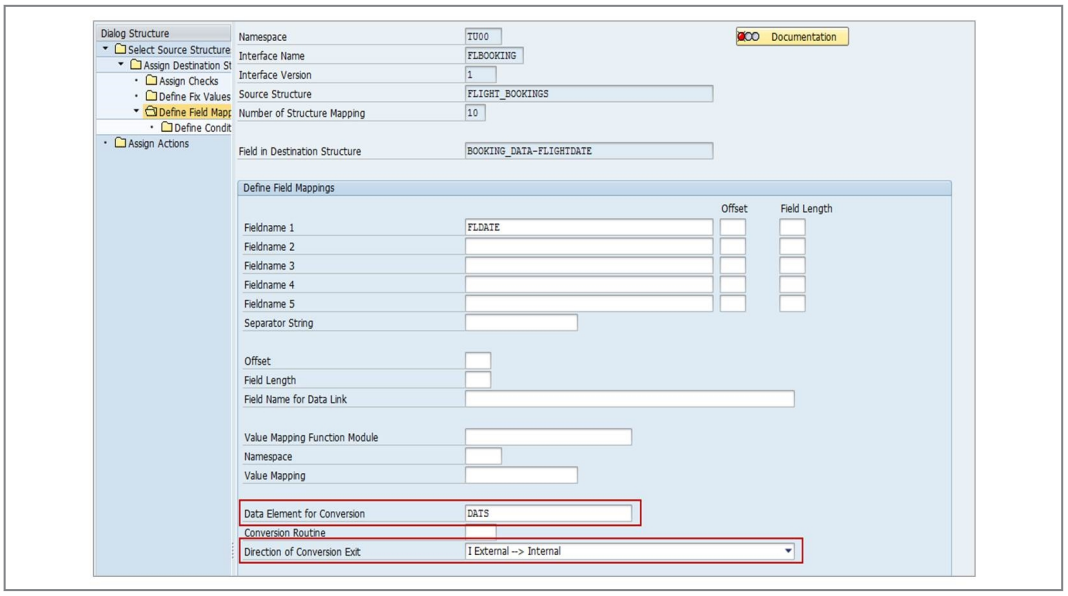
d) Navigate to *Define Field Mappings* and create a new entry for *FLIGHTDATE*.

e) Enter the data from the Destination Structure table.

f) Enter the data from the Define Field Mappings table.

> Note:
> To avoid the system ignoring the conversion data element, select the direction of conversion.

g) Save.



2. Test the value conversion.

a) Navigate to the AIF interface test tool (transaction `/AIF/IFTEST`).

b) Open your test file and transform.

c) In the SAP structure, fill out the FLIGHTDATE field. The flight date displays in format YYYYMMDD.

d) Process the data using Process in XML Runtime.

e) Open the Interface Monitor (transaction `/AIF/IFMON`).

f) Select the icon  .
You are forwarded to the Monitoring and Error Handling View.

g) Check the data.

> Note:
> In the AIF Monitoring and Error Handling View, the date displays in the correct format (DD.MM.YYYY). This is because the ALV table also converts the data before displaying it.

Unit 4
Exercise 7

Create a Value Mapping

Business Scenario

We want to create a value mapping for the fields *AIRLINEID* and *CONNECTIONID* of the *BOOKING_DATA* structure.

The content for *AIRLINEID* is the field *CARRID* of the *SFLIGHTS* table. It is read through the interface fields *AIRPORT_FROM*, *AIRPORT_TO*, and the already translated field *FLIGHTDATE*.

Read fields from a database table. Map a value with a value list for the field *CUSTOMERTYPE* . Create and assign a reusable value mapping *VM_AIRLINEID*. Create a field mapping for *AIRLINEID* using *AIRPORT_FROM*, *AIRPORT_TO*, and the translated *BOOKING_DATA-FLIGHTDATE*.

Create a field mapping for the AIRLINEID using the fields AIRPORT_FROM, AIRPORT_TO and the translated field BOOKING_DATA-FLIGHTDATE .

Note:
This exercise depends on completing the exercise **Map Structure Indirectly For Flight Bookings With Value Conversion**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Map a Value

1. Determine the *AIRLINEID*.
   Use the following data:

Field	Value
Field Name 1	**`AIRPORT_FROM`**
Field Name 2	**`AIRPORT_TO`**
Field Name 3	**`@BOOKING_DATA-FLIGHTDATE`**

2. Create the value mapping from the AIRLINEID's field mapping to derive the AIRLINEID.
   Use the following data:

Field	Value
CARRID	
Table Name	**`SFLIGHTS`**

Field	Value
Field Name	**CARRID**
Where Condition for Select Statement	

3. Build the Where condition.

   Use the following code:

```
AIRPFROM = '$1' AND AIRPTO = '$2' AND FLDATE = '$3'.
```

   Use the following data:

Field	Value
Single or Multiple Value Mapping	none
Customizing or Master Data	none

4. Optional step: Test the *AIRLINEID*.

Task 2: Determine the ConnectID

We also need a field mapping for the CONNECTID. The CONNECTID is determined out of the AIRPORT_FROM and AIRPORT_TO fields of the FLIGHT_BOOKINGS structure and the already mapped fields FLIGHTDATE and AIRLINEID of the BOOKING_DATA structure. The name of the Value mapping is VM_CONNECTID.

1. Determine the *CONNECTID*.

   Use the following data:

Field	Value
Field Name 1	**AIRPORT_FROM**
Field Name 2	**AIRPORT_TO**
Field Name 3	**@BOOKING_DATA-FLIGHTDATE**
Field Name 4	**@BOOKING_DATA-AIRLINEID**

2. Create a value mapping for CONNECTID.

   The connection ID will be derived by a value mapping.

   Use the following data:

Field	Value
new value mapping name	VM_CONNECTID
Table name	SFLIGHTS
Field Name	CONNID

3. Create the Where condition.

   Use the following code:`AIRPFROM = '$1' AND AIRPTO = '$2' AND FLDATE = '$3' AND CARRID = '$4'.` Use the following data:

Field	Value
Single or Multiple Value Mapping	none

Field	Value
Customizing or Master Data	none

4. Optional step: Test the *CONNECTID*.

Task 3: Determine the Customer Type

The customer type comes in the source interface but it has more than one characters and has to be translated to the one-character field that is needed in the target structure.

The word Company shall be translated to B, the word Standard to P. The Data Dictionary Data Element S_CUSTTYPE can be used for the value table.

1. Define Value Mapping.

   Use the following data:

Field	Value
Name	VM_CUSTOMERTYPE
Description	<free text>
Data Element for INT	S_CUSTTYPE
Number of External Values	1
Single or Multiple Value Mapping	Single
Customizing or Master Data	Customizing

2. Maintain Value Mapping.

   Use the following data:

Index	Ext. Value	Internal Value
1	Company	B

3. Add Value Mapping to Field Mapping.
4. Test Value Mapping.

   The new mapping can be tested in the AIF test tool.

5. Maintain Missing Value Mapping.

   The missing entry can be added in `/AIF/VMAP` as before. But it also can be maintained directly from the AIF Error handling. Add the missing entry **`P`** for *Standard*.

   Use the following data:

Index	Ext. Value	B/P Customer
2	Standard	P

Task 4: Link Data for AIRLINEID and CONNECTID

1. Create a data link.

   Use the following data:

Field Mapping	Field Name for Data Link
AIRLINEID	**FLDATE**
CONNECTID	**FLDATE**

2. Test the data link.
   Use the following data:
   FLDATE: **99999999**

Unit 4
Solution 7

Create a Value Mapping

Business Scenario

We want to create a value mapping for the fields *AIRLINEID* and *CONNECTIONID* of the *BOOKING_DATA* structure.

The content for *AIRLINEID* is the field *CARRID* of the *SFLIGHTS* table. It is read through the interface fields *AIRPORT_FROM*, *AIRPORT_TO*, and the already translated field *FLIGHTDATE*.

Read fields from a database table. Map a value with a value list for the field *CUSTOMERTYPE* . Create and assign a reusable value mapping *VM_AIRLINEID*. Create a field mapping for *AIRLINEID* using *AIRPORT_FROM*, *AIRPORT_TO*, and the translated *BOOKING_DATA-FLIGHTDATE*.

Create a field mapping for the AIRLINEID using the fields AIRPORT_FROM, AIRPORT_TO and the translated field BOOKING_DATA-FLIGHTDATE .

Note:
This exercise depends on completing the exercise **Map Structure Indirectly For Flight Bookings With Value Conversion**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Map a Value

1. Determine the *AIRLINEID*.

   Use the following data:

Field	Value
Field Name 1	**`AIRPORT_FROM`**
Field Name 2	**`AIRPORT_TO`**
Field Name 3	**`@BOOKING_DATA-FLIGHTDATE`**

   a) Navigate to *AIF Customizing (transaction /AIF/CUST)* → *Interface Development* → *Define Structure Mappings*.

   b) Choose *Define Field Mappings* for *FLIGHT_BOOKINGS*.

   c) To create a field mapping for *BOOKING_DATA-AIRLINEID*, enter the airline ID through the departure airport, the destination airport, and the flight date.

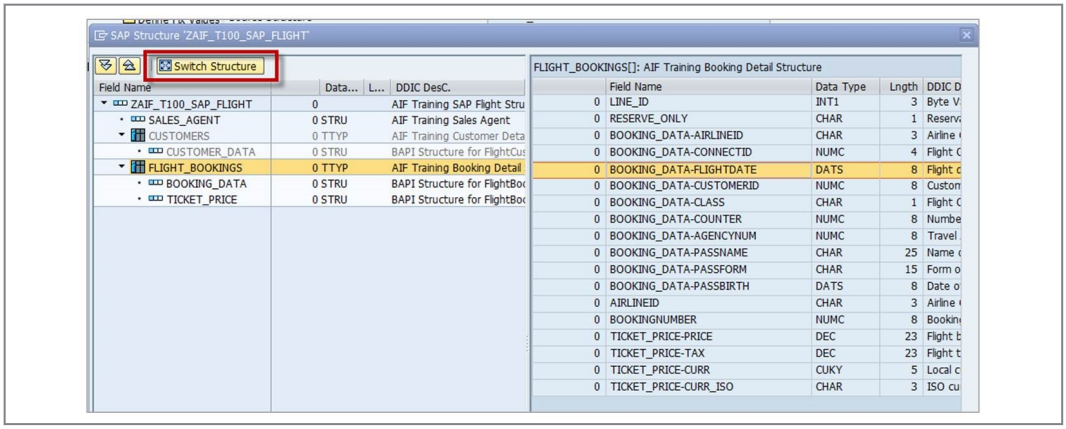
The correct AIRLINEID's is determined through a value mapping.

d) Using the parameters specified in the Field Names table, complete the information in the value mapping.

> Note:
> The '@' sign in Field Name 3 indicates the use of a field from the SAP data structure. You can enter the name of the field manually or you can use the value help.

e) In the dialog, select *SWITCH STRUCTURE*.
The SAP structure displays.

f) Choose the *FLIGHTDATE* and confirm.
The '@' sign is automatically added as a prefix.



g) Save.

2. Create the value mapping from the AIRLINEID's field mapping to derive the AIRLINEID.
Use the following data:

Field	Value
CARRID	
Table Name	**SFLIGHTS**
Field Name	**CARRID**
Where Condition for Select Statement	

a) In the *Define Field Mappings*, enter your namespace into the *Namespace* field.

b) As name for the value mapping you want to create enter *VM_AIRLINEID*.

c) Choose *Enter*.
You will get an error.

d) Choose *Enter* again and *Confirm*.

e) Save.

f) To start defining value mapping, double click on the value mapping's name.

g) Switch to change mode and enter a description.

> Note:
> Flight data is contained in the *SFLIGHTS* table.

h) Enter the data from the table above.

CARRID corresponds to *AIRLINEID* in table *SFLIGHTS*. You can also use the value help of the field.

3. Build the Where condition.

Use the following code:

```
AIRPFROM = '$1' AND AIRPTO = '$2' AND FLDATE = '$3'.
```

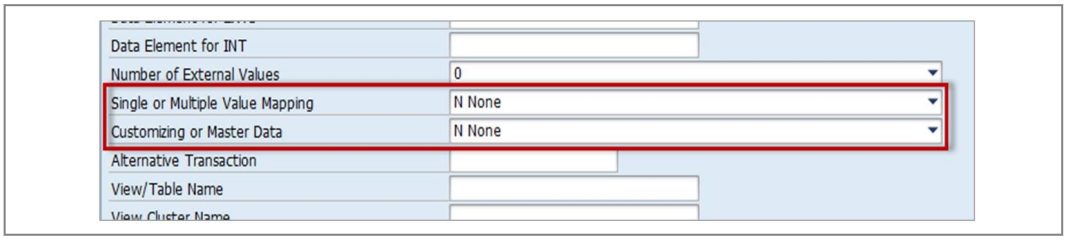
Use the following data:

Field	Value
Single or Multiple Value Mapping	none
Customizing or Master Data	none

a) To build the WHERE condition, enter the data above into the *Where Condition for Select Statement* field.

> Note:
> The parameter values **$1**, **$2** , and **$3** are replaced at runtime by the values entered in *Fieldname 1*, *Fieldname 2*, and *Fieldname 3*.

b) Under *Single or Multiple Value Mapping* and *Customizing or Master Data*, select **none**.



c) Save and close the window.

4. Optional step: Test the *AIRLINEID*.

a) Navigate to *AIF interface* test tool (transaction **/AIF/IFTEST**) and open your test file.

b) Choose *Transform*.

c) Check the results in the SAP structure.
If the mapping succeeds, the *AIRLINEID* of the *BOOKING_DATA* substructure fills. If an error occurs during mapping, a message displays in the *Display Logs* section, that there is no result for specified selection in table SFLIGHTS.

Task 2: Determine the ConnectID

We also need a field mapping for the CONNECTID. The CONNECTID is determined out of the AIRPORT_FROM and AIRPORT_TO fields of the FLIGHT_BOOKINGS structure and the already mapped fields FLIGHTDATE and AIRLINEID of the BOOKING_DATA structure. The name of the Value mapping is VM_CONNECTID.

1. Determine the *CONNECTID*.

   Use the following data:

Field	Value
Field Name 1	**AIRPORT_FROM**
Field Name 2	**AIRPORT_TO**
Field Name 3	**@BOOKING_DATA-FLIGHTDATE**
Field Name 4	**@BOOKING_DATA-AIRLINEID**

   a) Navigate to *AIF Customizing (transaction /AIF/CUST)* → *Interface Development* → *Define Structure Mappings*.

   b) Select *FLIGHT_BOOKINGS* and navigate to Define Field Mappings on the left hand side.

   c) Create a field mapping for *BOOKING_DATA-CONNECTID*.

   d) Enter the field name directly in Field in Destination Structure or use the value help.

   e) Enter the data from the Field in Destination Structure table.

   The '@' sign in *Field Name 3* and *Field Name 4* indicates that a field from the SAP data structure is used. You can enter the name of the field manually or you can use the value help.

   f) In the dialog, select *SWITCH STRUCTURE* in the top left corner.

   The SAP structure displays.

   g) For *Field Name 3*, choose *FLIGHTDATE* and, for *Field Name 4*, choose *AIRLINEID*, and confirm.

   Ensure you select the AIRLINEID from the *BOOKING_DATA* structure. The '@' sign is automatically added as a prefix.

2. Create a value mapping for CONNECTID.

   The connection ID will be derived by a value mapping.

   Use the following data:

Field	Value
new value mapping name	VM_CONNECTID
Table name	SFLIGHTS
Field Name	CONNID

   a) To create the value mapping, enter your namespace and a new value mapping name, into the *Value Mapping* field in *Define Field Mapping for the connection ID*.

b) A pop-up displays. Confirm that you want to create the new value mapping.

c) Save.

> Note:
> If an error message informs you that the value mapping does not exist, press *Enter* again. The pop-up displays. Confirm that you want to create the value mapping and save.

d) Double click on the value mapping's name to jump into the value mapping and switch to change mode. Enter a description.

e) Flight data is contained in the SFLIGHTS table. Enter **`SFLIGHTS`** in the *Table Name* field. CONNID is the corresponding field to CONNECTID in the SFLIGHTS table. Enter **`CONNID`** in the *Field Name* field (or use the value help of the field).

3. Create the Where condition.

   Use the following code:`AIRPFROM = '$1' AND AIRPTO = '$2' AND FLDATE = '$3' AND CARRID = '$4'`.Use the following data:

Field	Value
Single or Multiple Value Mapping	none
Customizing or Master Data	none

   a) To build the WHERE condition, enter the data above into the *Where Condition for Select Statement* field.

   b) Under *Single or Multiple Value Mapping* and *Customizing or Master Data*, select **`none`**.

4. Optional step: Test the *CONNECTID*.

   a) Navigate to *AIF interface* test tool (transaction **`/AIF/IFTEST`**) and open your test file.

   b) Choose *Transform*.

   c) Check the results in the SAP structure.
   The *CONNECTID* field of the *BOOKING_DATA* substructure should be filled if the mapping was successful. If an error occurred during the mapping, a message **`no result for specified selection in table SFLIGHTS`** displays in the *Display Logs* section of the screen.

Task 3: Determine the Customer Type

The customer type comes in the source interface but it has more than one characters and has to be translated to the one-character field that is needed in the target structure.

The word Company shall be translated to B, the word Standard to P. The Data Dictionary Data Element S_CUSTTYPE can be used for the value table.

1. Define Value Mapping.

   Use the following data:

Field	Value
Name	VM_CUSTOMERTYPE

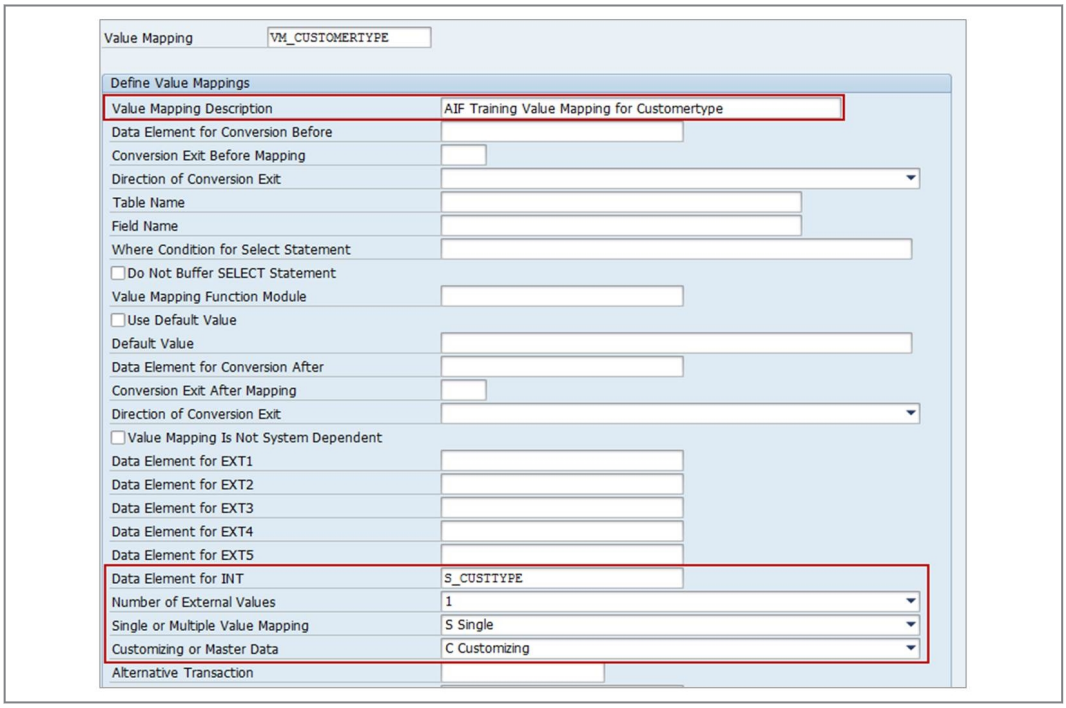
Field	Value
Description	<free text>
Data Element for INT	S_CUSTTYPE
Number of External Values	1
Single or Multiple Value Mapping	Single
Customizing or Master Data	Customizing

a) Go to AIF Customizing (transaction `/AIF/CUST`).

b) Under *Interface Development → Define Value Mappings* select your namespace.

c) Create a new entry; enter a name and a description for the Value Mapping.

d) Enter the information from the table above in the corresponding fields.



e) Save.

2. Maintain Value Mapping.

Use the following data:

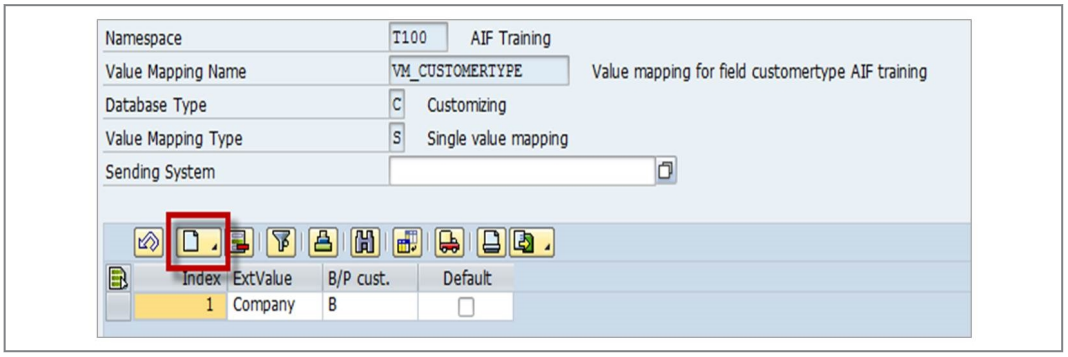
Index	Ext. Value	Internal Value
1	Company	B

a) Go to transaction `/AIF/VMAP` – Maintenance of Value Mappings.

b) Enter your namespace and select the previously created Value Mapping with the F4 Help. Press *Execute*.

c) Switch to edit mode on the next screen and create a new entry via the *Append* button.

d) Enter the data from the table above.



3. Add Value Mapping to Field Mapping.

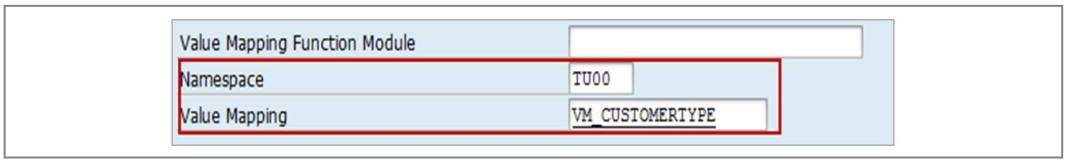
a) Go to AIF Customizing (transaction */AIF/CUST*).

b) Under *Interface Development → Define Structure Mappings* and enter your namespace and interface.

c) As *Source Structure* select *CUSTOMER_DATA* and navigate to *Define Field Mappings* on the left hand side of the screen.

d) Open the field mapping for field *CUSTTYPE*. In the *Namespace* field above the field *Value Mapping*, enter your namespace. In the Value Mapping field, enter your created value mapping (value help is available).

e) Save.



4. Test Value Mapping.

The new mapping can be tested in the AIF test tool.

a) Execute transaction `/AIF/IFTEST`.

b) Open your test file and select *Transform*.

c) Check the result in the SAP structure.

In the *CUSTOMER_DATA* structure, the *CUSTTYPE* should be changed. Instead of **`Company`** the field should contain **`B`**.

d) Also, you should see an error message telling you that the value Standard does not exist in the value mapping.

5. Maintain Missing Value Mapping.

The missing entry can be added in `/AIF/VMAP` as before. But it also can be maintained directly from the AIF Error handling. Add the missing entry **`P`** for *Standard*.

Use the following data:

Index	Ext. Value	B/P Customer
2	Standard	P

a) In `/AIF/IFTEST` process your testfile via the XML runtime.

b) Open the message in the error handling.

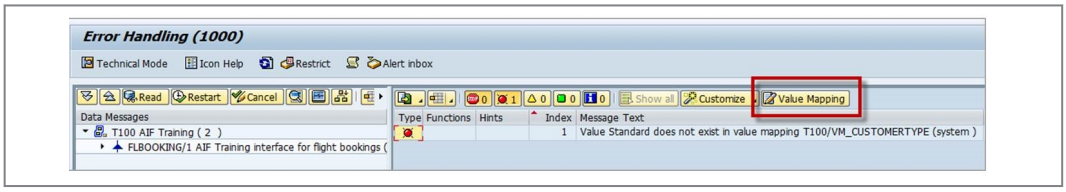
c) Open the Interface Monitor via transaction `/AIF/IFMON`.

d) Select the icon .

You are forwarded to the *Monitoring and Error Handling* view.

e) Check the data.

f) The message should be displayed in error status. To remove the error, you have to maintain the missing value mapping



g) Select your message and click on the *Value Mapping* button.

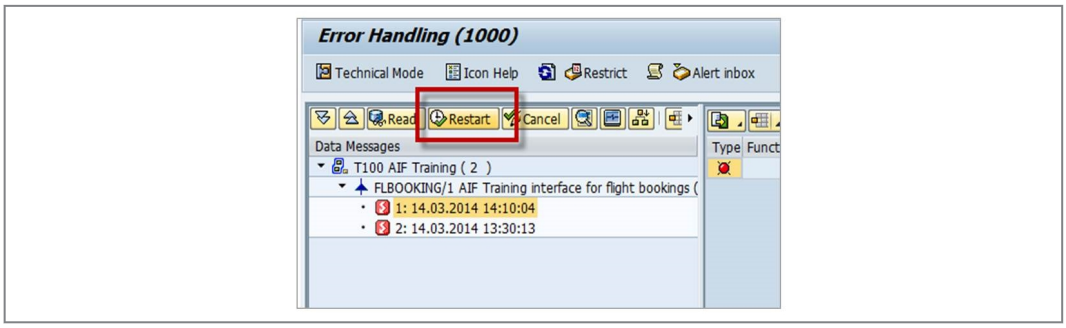
The maintenance view for the value mappings opens.

h) Create a new entry with the information from the table above.:

i) Save the changes

j) Navigate back to *Monitoring and Error Handling*.

k) Select your message and press the *Restart* button.



l) Check the result of the message.

The message is processed successfully.

Task 4: Link Data for AIRLINEID and CONNECTID

1. Create a data link.

Use the following data:

Field Mapping	Field Name for Data Link
AIRLINEID	**FLDATE**
CONNECTID	**FLDATE**

a) Navigate to *Interface Development → Define Structure Mappings → AIF Customizing (transaction /AIF/CUST)*.

b) Select your namespace and interface.

c) Select the *FLIGHT_BOOKINGS* source structure and navigate to the field mappings.

d) For each field mapping entry in the table, enter the data from the table, and save.
If necessary, use the value help to select the correct field.

2. Test the data link.
Use the following data:

FLDATE: **99999999**

a) Navigate to the AIF interface test tool (transaction `/AIF/IFTEST`) and open your test file.

b) Navigate into the FLIGHT_BOOKINGS structure of the raw structure.

c) Change *FLDATE* of one entry to the value provided in the step.

d) Save.

e) Process in XML Runtime.

f) Open Monitor and Error Handling (`/AIF/ERR`).

g) Select your namespace and interface.

h) Select all messages from today.

i) Display the newest message.

j) Double click the error message.
In the data content view at the bottom of the screen, the erroneous entry in the *FLDATE* field is highlighted.

Unit 4
Exercise 8

Create a Conditional Mapping

Business Scenario

You want to fill the CLASS that is needed in the target interface.

Fill it in the BOOKING_DATA or in the CUSTOMER_DATA structure of the source interface. If it exists in the BOOKING_DATA structure, use this field. If it is empty, use the CUSTOMER_DATA field.

Note:
This exercise depends on completing the exercise **Create a Value Mapping**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Define and Test a Conditional Mapping

1. Define Conditional Mapping.

   Use the following data:

Field	Value
Condition Number	`10`
Alternative Value Mapping / Field Name	Selected
Field Check	`I Empty`
Fieldname 1	`CLASS`
Alternative Fieldname 1	`CUSTOMER_DATA!CUSTOMER_DETAILS-CLASS`

2. Test Your Mapping.

   Use the following data:

SAP Structure Element	Details
FLIGHT_BOOKINGS	Contains rows
Field class	Every entry filled
Customer 1 - First entry	*CLASS*: `Y`
Customer 1 - Second entry	*CLASS*: `C`

SAP Structure Element	Details
Customer 2	*CLASS*: **`F`**
Other *CLASS* entries in *FLIGHT_BOOKING*	Empty

Unit 4
Solution 8

Create a Conditional Mapping

Business Scenario

You want to fill the CLASS that is needed in the target interface.

Fill it in the BOOKING_DATA or in the CUSTOMER_DATA structure of the source interface. If it exists in the BOOKING_DATA structure, use this field. If it is empty, use the CUSTOMER_DATA field.

Note:
This exercise depends on completing the exercise **Create a Value Mapping**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Define and Test a Conditional Mapping

1. Define Conditional Mapping.

   Use the following data:

Field	Value
Condition Number	`10`
Alternative Value Mapping / Field Name	Selected
Field Check	`I Empty`
Fieldname 1	`CLASS`
Alternative Fieldname 1	`CUSTOMER_DATA!CUSTOMER_DETAILS-CLASS`

   a) Navigate to *Interface Development → Define Structure Mappings → AIF Customizing (transaction /AIF/CUST)*.

   b) Select *FLIGHT_BOOKINGS* as source structure and navigate to *Define Field Mappings*.

   c) Select field mapping for the *CLASS* field and choose *Define Conditions*.

   d) To create a new entry, enter the data provided in the step.

   To select the values for *Fieldname 1* and *Alternative Fieldname 1*, use the value help.

   e) Save.

2. Test Your Mapping.

   Use the following data:

SAP Structure Element	Details
FLIGHT_BOOKINGS	Contains rows
Field class	Every entry filled
Customer 1 - First entry	*CLASS*: `Y`
Customer 1 - Second entry	*CLASS*: `C`
Customer 2	*CLASS*: `F`
Other *CLASS* entries in *FLIGHT_BOOKING*	Empty

   a) Open the AIF interface test tool (/AIF/IFTEST) and select your file.

   b) Process the test file.

   c) Using the data provided in the step, check the results in the SAP structure.

Unit 5
Exercise 9

Create Actions

Business Scenario

You want to post customer data and flight bookings. Create a new customer for every entry in the CUSTOMERS structure. To post the customers, you can use the BAPI `BAPI_FLCUST_CREATEFROMDATA`. The BAPI already exists in the system and does exactly what you need.

After you create the customer, you want to post the corresponding flights.

Note:
This exercise depends on completing the exercise **Create A Conditional Mapping**.

Note:
Before beginning this exercise, ensure you leave all other transactions.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create a New Customer

1. Define Action AC_FLCUST_CREATE.
   Use the following data:

Field	Value
Action number	10
Record Type	CUSTOMERS
Change Mode	
Commit Mode	**`COMMIT WORK AND WAIT`**
Commit Level	**`AFTER EACH FUNCTION`**
Main Component Type	**`ZAIF_T100_CUSTOMER_DETAIL`**

2. Use AC_FLCUST_CREATE for your interface.
   Use the following data:

Field	Value
Function	
Function Number	**10**
Function module	**ZAIF_S###_AC_FLCUST_CREATE**
Package and Workbench Request	
Function group	**ZAIF_S###**
Package	**ZAIF_S###**
Change Parameters	
Associated Type for CURR_LINE	**ZAIF_T100_CUSTOMER_DETAIL**

3. Test your action.
   Use the AIF test tool.

Task 2: Post Flight Bookings
After creating the customer creation, you want to post the corresponding flights. You want to create a second action AC_FLBOOKING_POST for the flight postings. The BAPI `BAPI_FLBOOKING_CREATEFROMDATA` already exists to do that.

1. Define action AC_FLBOOKING_POST and use it in your interface.
   Use the following data:

Field	Value
Action	
Action number	**20**
Action	**AC_FLBOOKING_POST**
Function	
Commit Mode	**COMMIT WORK AND WAIT**
Commit Level	**AFTER EACH FUNCTION**
Main Component Type	**ZAIF_T100_BOOKING_DETAIL**
New Definition	
New entry	Number 10
Function module name	**ZAIF_S###_AC_FLBOOKING_POST**
Stop Processing on Error	Blank
Restart Always	Blank
Record Type	FLIGHT_BOOKINGS

2. Implement the action function module.

Note:
If you are not on the function module overview screen, click the Back button once to go back. Double-click on the Function Module that was just created. A new window opens.

In the changing parameters of the function module, **`CURR_LINE`** is specified. Change the *Associated Type* to the line type of the bookings table type *ZAIF_T100_BOOKING_DETAIL*. You do not need this step for AIF to work It simplifies the function module's implementation. The syntax check already determines access to a field in the non-existent structure. It is not as a dump at runtime.

3. Test your action.

Unit 5
Solution 9

Create Actions

Business Scenario

You want to post customer data and flight bookings. Create a new customer for every entry in the CUSTOMERS structure. To post the customers, you can use the BAPI `BAPI_FLCUST_CREATEFROMDATA`. The BAPI already exists in the system and does exactly what you need.

After you create the customer, you want to post the corresponding flights.

Note:
This exercise depends on completing the exercise **Create A Conditional Mapping**.

Note:
Before beginning this exercise, ensure you leave all other transactions.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create a New Customer

1. Define Action AC_FLCUST_CREATE.

   Use the following data:

Field	Value
Action number	10
Record Type	CUSTOMERS
Change Mode	
Commit Mode	**`COMMIT WORK AND WAIT`**
Commit Level	**`AFTER EACH FUNCTION`**
Main Component Type	**`ZAIF_T100_CUSTOMER_DETAIL`**

   a) If open, close all windows to all different systems, except one.

   b) In this window, leave all transactions.

c) Navigate to *Interface Development → Define Structure Mappings → AIF Customizing (transaction /AIF/CUST)*.

d) Select your namespace, interface, and version.

e) Choose *Assign Actions*.

f) To create a new entry, enter the action number provided in the step.

g) Enter your namespace and a name for the action (`AC_FLCUST_CREATE`).

h) Choose *Enter* and confirm that you want to create the action.

i) Enter the *Record Type* provided in the step.
Enter value directly or per value help.

j) Save.

k) Use forward navigation (double click on the action name) to navigate to *Define Actions*.

l) Navigate to the action for new customers.

m) Switch to change mode and enter a description.

n) Enter the data provided in the Change Mode table.

2. Use AC_FLCUST_CREATE for your interface.
Use the following data:

Field	Value
Function	
Function Number	**10**
Function module	**ZAIF_S###_AC_FLCUST_CREATE**
Package and Workbench Request	
Function group	**ZAIF_S###**
Package	**ZAIF_S###**
Change Parameters	
Associated Type for CURR_LINE	**ZAIF_T100_CUSTOMER_DETAIL**

a) Navigate to *Define Functions*.

b) To create a new entry, enter data provided in the Function table.

c) Choose *Enter*.

d) Confirm that you want to create the function module.

e) Enter the data provided in the Package and Workbench Requestion table.

f) Save.

g) To start creating the function module, jump into the created function module.

h) Activate change mode.

i) Navigate to the changing parameters and enter the data provided in the Change Parameters table.

j) Insert the following code into the function module:

```
CALL FUNCTION 'BAPI_FLCUST_CREATEFROMDATA'
       EXPORTING
          customer_data = curr_line-CUSTOMER_DATA
       IMPORTING
         customernumber = curr_line-id
       TABLES
         return = return_tab.
```

k) Save and activate.

3. Test your action.

Use the AIF test tool.

a) Navigate to the AIF interface test tool (transaction `/AIF/IFTEST`) and open your file.

Note:
Ensure that you restart the test transaction to ensure that the newly created action function module can be executed.

b) Execute the test using *Process in XML Runtime*.

c) Navigate to the Interface Monitor (transaction `/AIF/IFMON`).

d) Select your namespace and interface and select all the messages from today.

e) Open the newest message.
The message is processed successfully. The log messages state that a new customer is created.

Task 2: Post Flight Bookings

After creating the customer creation, you want to post the corresponding flights. You want to create a second action AC_FLBOOKING_POST for the flight postings. The BAPI `BAPI_FLBOOKING_CREATEFROMDATA` already exists to do that.

1. Define action AC_FLBOOKING_POST and use it in your interface.

Use the following data:

Field	Value
Action	
Action number	**20**
Action	**AC_FLBOOKING_POST**
Function	
Commit Mode	**COMMIT WORK AND WAIT**
Commit Level	**AFTER EACH FUNCTION**
Main Component Type	**ZAIF_T100_BOOKING_DETAIL**
New Definition	

Field	Value
New entry	Number 10
Function module name	**ZAIF_S###_AC_FLBOOKING_POST**
Stop Processing on Error	Blank
Restart Always	Blank
Record Type	FLIGHT_BOOKINGS

a) Navigate to *Interface Development* → *Define Structure Mappings* → *AIF Customizing (transaction /AIF/CUST)*.

b) Select *Assign Actions*.

c) To create a new action, enter the data provided in the Action table.

d) Choose *Enter* and confirm that the action should be created.

e) Select the *Record Type* value provided in the step.
Use the value help to do so.

f) Save.

g) Define a function.
Use forward navigation to jump to your action and switch to change mode.

h) Enter a description and the data provided in the Function table.

i) Proceed to the define functions view.

j) To create a new entry, enter the data provided in the New Definition table.

k) To create the function module, choose *Enter* or *Save* and confirm the dialog. The system automatically determines the correct function module template to be used.

l) Enter the name of your ZAIF_S### function group and save.

m) Confirm the creation of the function group and choose your ZAIF_S### package.

2. Implement the action function module.

Note:
If you are not on the function module overview screen, click the Back button once to go back. Double-click on the Function Module that was just created. A new window opens.

In the changing parameters of the function module, **CURR_LINE** is specified. Change the *Associated Type* to the line type of the bookings table type *ZAIF_T100_BOOKING_DETAIL*. You do not need this step for AIF to work It simplifies the function module's implementation. The syntax check already determines access to a field in the non-existent structure. It is not as a dump at runtime.

a) Insert the following coding in the function module:

```
CALL FUNCTION 'BAPI_FLBOOKING_CREATEFROMDATA'
        EXPORTING
            reserve_only = curr_line-reserve_only
            booking_data = curr_line-booking_data
        IMPORTING
            airlineid = curr_line-airlineid
            bookingnumber = curr_line-bookingnumber
            ticket_price = curr_line-ticket_price
        TABLES
            return = return_tab
```

b) Save and activate the function module. Save the changes in the other windows.

3. Test your action.

a) Navigate to AIF interface test tool (transaction `/AIF/IFTEST`) and open your file.

b) Process the file via the XML runtime.
The message ends in error state. The error is because the flight booking does not know the customer number of the newly created customer.

Unit 5
Exercise 10

Use Additional Action Features

Business Scenario

You want to create a new customer before executing the flight booking. You must save the customer first and use the new customer ID to book the flights. In the case of an error during the booking, you do not want the customer to be created again. Use a field to store the new customer ID. When posting a flight booking fails because of a wrong flight class, you want to edit this field before trying to post again. You make the FLIGHTBOOKING-CLASS field editable.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Note:
This exercise depends on completing the exercise **Create Actions**.

Task 1: Link the New Customer to the Flight Bookings

Link the newly created customers to the corresponding flight bookings. You must create a function that is processed before the flight booking action is processed. In this function, map the new customer ID to the corresponding flight bookings.

1. Create a function before processing.

   Use the following data:

Field	Value
Action	AC_FLBOOKING_POST
Init Function before Processing	
Name	ZAIF_S###_INIT_AC_FLBOOKING
DATA	ZAIF_S###_SAP_FLIGHT

2. Test an Init Function before processing.

Task 2: Create Fields to Restore

The customer ID is returned from the BAPI that creates customers. It is inserted into the SAP structure in the *ZAIF_S###_AC_FLCUST_CREATE* action function module. The customer ID is available in the second action, *AC_FLBOOKING_POST*, and the corresponding field in the flight booking is editable.

1. Create fields to restore.

2. Test your field to restore.

   Use the following data:

Field	Value
Flight_bookings-class	G

Task 3: Correct Errors

The last message that was sent had an error with an incorrect flight class. You want to be able to correct such errors. Therefore, the field containing the flight class has to be set as editable.

1. Create a changeable field for flight class.

   Use the following data:

Field	Value
Index	10

2. Test the changeable fields and reprocessing.

   Use the following data:

Field	Value
Valid flight class	Y
Data Messages	Restart

Unit 5
Solution 10

Use Additional Action Features

Business Scenario

You want to create a new customer before executing the flight booking. You must save the customer first and use the new customer ID to book the flights. In the case of an error during the booking, you do not want the customer to be created again. Use a field to store the new customer ID. When posting a flight booking fails because of a wrong flight class, you want to edit this field before trying to post again. You make the FLIGHTBOOKING-CLASS field editable.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Note:
This exercise depends on completing the exercise **Create Actions**.

Task 1: Link the New Customer to the Flight Bookings

Link the newly created customers to the corresponding flight bookings. You must create a function that is processed before the flight booking action is processed. In this function, map the new customer ID to the corresponding flight bookings.

1. Create a function before processing.

   Use the following data:

Field	Value
Action	AC_FLBOOKING_POST
Init Function before Processing	
Name	ZAIF_S###_INIT_AC_FLBOOKING
DATA	ZAIF_S###_SAP_FLIGHT

   a) Navigate to *Interface Development → Define Actions → AIF Customizing (transaction /AIF/CUST)*.

   b) Select your namespace.
   You have an overview of all actions defined for your namespace.

   c) Switch to change mode.

   d) Double click on the action for the flight booking (**`AC_FLBOOKING_POST`**).

e) In *Init Function before Processing*, enter a name for the *ZAIF_S###_INIT_AC_FLBOOKING* function.

f) Choose *Enter* and confirm that you want to create the function module.

g) Enter your ZAIF_S###. Function Group.

h) Save.

i) Navigate into the function module.

j) Switch to change mode.

k) Navigate to the changing tab and change the Associated Type of parameter *DATA* to **`ZAIF_S###_SAP_FLIGHT`**.

l) Insert the following source code:

```
FUNCTION zaif_s###_init_ac_flbooking .
  DATA: ls_customer TYPE zaif_t100_customer_detail.

  FIELD-SYMBOLS: <ls_flight_data> TYPE zaif_t100_booking_detail.

  LOOP AT data-flight_bookings ASSIGNING <ls_flight_data>.
    READ TABLE data-customers INTO ls_customer INDEX <ls_flight_data>-
line_id.
    IF sy-subrc = 0.
      <ls_flight_data>-booking_data-customerid = ls_customer-id.
    ENDIF.
  ENDLOOP.
ENDFUNCTION.
```

2. Test an Init Function before processing.

a) Navigate to the AIF interface test tool (transaction `/AIF/IFTEST`) and open your test file.

b) Use Process in XML runtime.

c) Open the Interface Monitor via transaction `/AIF/IFMON`.

d) Select the icon.
You are forwarded to *Monitoring and Error Handling*.

e) Check the results.

f) Click on the message and check the entries in *Log Messages*.
Customer and flight bookings have been created successfully.

Task 2: Create Fields to Restore

The customer ID is returned from the BAPI that creates customers. It is inserted into the SAP structure in the *ZAIF_S###_AC_FLCUST_CREATE* action function module. The customer ID is available in the second action, *AC_FLBOOKING_POST*, and the corresponding field in the flight booking is editable.

1. Create fields to restore.

a) Navigate to *Interface Development → Define Actions → AIF Customizing (transaction /AIF/CUST)*.

b) Enter your namespace.

c) To generate your customer (**`AC_FLCUST_CREATE`**), select your created action.

d) Choose *Define Functions*.

e) Select the function that creates the customer.

f) Choose *Define Fields to Restore*.

g) Create a new entry.

h) Enter ID as field name.

i) Save.

2. Test your field to restore.
   Use the following data:

Field	Value
Flight_bookings-class	G

a) Navigate to the AIF interface test tool (transaction `/AIF/IFTEST`) and open your test file.

b) To prevent a flight booking posting successfully, for an entry in *FLIGHT_BOOKINGS*, enter the non-existent flight class provided in the step.

c) Save the changes and execute the test file using Process in XML Runtime.

d) Execute transaction `/AIF/IFMON`.

e) To check the results, navigate to Monitoring and Error Handling.

f) Select the most recent message.
There is an error message that the flight class can only be Y, C, or F.

g) In *Log Messages*, to display the success log messages, select the green status icon ().
You see messages that your customers have been created.

Task 3: Correct Errors

The last message that was sent had an error with an incorrect flight class. You want to be able to correct such errors. Therefore, the field containing the flight class has to be set as editable.

1. Create a changeable field for flight class.
   Use the following data:

Field	Value
Index	10

a) Execute transaction `/AIF/CUST` and navigate to *Error Handling → Namespace-Specific Features*.

b) Enter your namespace.

c) In *Define Namespace-Specific Features*, choose *New Entries*.

d) Save and return to *Customizing of the SAP Application Interface Framework*.

e) d. Navigate to *Error Handling → Interface-Specific Features*.

f) Select *Define Changeable Fields*.

g) Choose *New Entries*.

h) Enter the index value **10**.

i) Select the flight class from the F4 help.

Note:
The raw structure contains two flight classes. Ensure you select the flight class from the *FLIGHT_BOOKINGS* structure. *Field Path* contains *CUSTOMER_DATA-FLIGHT_BOOKINGS-CLASS*.

j) Save.

2. Test the changeable fields and reprocessing.
Use the following data:

Field	Value
Valid flight class	Y
Data Messages	Restart

a) In the AIF interface test tool (transaction `/AIF/IFTEST`), open your test file and enter a non-existent flight class in *FLIGHT_BOOKINGS*.

b) Save your file and process the data in the XML runtime.

c) Navigate to the *Interface Monitor* and access *Monitoring and Error Handling*.

d) Select the newest message.
In *Log Messages*, you see an error that the flight could not be created due to an incorrect flight class.

e) Display the success messages in *Log Messages*.
You see messages that your customers were created.

f) Note the customer numbers of the newly created customers.

Note:
The numbers are required to check if the field for restoring functionality works correctly.

g) In *Data Structure*, select the *FLIGHT_BOOKINGS* table.
The content of the table displays in *Data Content*.

h) Double click on the field with the flight class.

i) In the dialog, enter the valid flight class provided in the step and choose *OK*.

j) In *Data Content*, choose *Save*.

k) To restart the message, in *Data Messages*, select the Data Messages value **Restart**.

l) Wait two seconds, then double click the data message.
The data message is successfully processed.

m) Check the success messages in *Log Messages*.
The success messages for the customer creation has not changed. The customer ID of the previous processing is used to create flight bookings.

Unit 6
Exercise 11

Create and Use Index Tables

Business Scenario

You want to help business users search for specific message content in *Monitoring and Error Handling*. To enable the search, create interface-specific index tables and an interface-specific selection screen. The index tables enable you to store the search fields' values. The interface-specific selection screen displays as an additional sub-selection screen on the selection screen of *Monitoring and Error Handling*.

Note:
This exercise depends on completing the exercise **Set Additional Actions**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create an Interface-Specific Single Index Table

1. Create a single index table.
   Use the following data:

Field	Data Element
Transaction	SE11
Source table	/AIF/STD_IDX_TBL
Target table	ZS###_FBOOK_SIDX
Package	ZAIF_S###
Add the following fields:	
AGENCY	**S_AGNCYNUM**
SALES_AGENT_ID	**ZAIF_T100_SALES_AGENT_ID**

Task 2: Create a Multi Index Table

To store values that can occur multiple times, create a multi index table in order. The values are contained in a structure that can exist more than once in the interface. For this interface, *AIRPORT_FROM*, *AIRPORT_TO*, and *CUSTOMERS-ID* can occur multiple times. Searches for those values require a multi index table. The basic version of a multi index table appears like this:

Field	Key Field	Initial Values	Data Element	Group
MANDT	X	X	MANDT	

Field	Key Field	Initial Values	Data Element	Group
MSGGUID	X	X	GUID_32	
COUNTER	X	X	INT4	
.INCLUDE			/AIF/IFKEYS	AIFKEYS
.INCLUDE			/AIF/ADMIN	ADMIN
PID			SXMSPID	

The table `/AIF/MIDX_TEMPL` is already a basic version of the multi index table found in the system. The two airport fields are from the source structure and the customerid from the target structure. They cannot be mixed in one index table. The customer ID requires a second index table because it is not related to the airport fields. Create two multi index tables.

1. Create the multi index table for the source structure.
   Use the following data:

Field	Value
Transaction	SE11
Source table	/AIF/MIDX_TEMPL
New table	ZS###_FBOOK_MIDX
Airport Fields	
AIRPORT_FROM	**S_FROMAIRP**
AIRPORT_TO	**S_TOAIRP**

2. Create the multi index table for customer ID.
   Use the following data:

Field	Value
Transaction	SE11
Source table	/AIF/MIDX_TEMPL
New table	ZS###_FCUST_MIDX
CUSTOMER	**S_CUSTOMER**

Task 3: Create an Interface-Specific Selection Screen

You need an interface-specific screen for selecting from both the single index and multi index tables. You need a module pool and must program the screen. For every field not part of the standard single index table, you need variables and select-options or parameters.

In your case, these are fields for agency, sales agent, two airport fields, and the customerid. The screen must be type subscreen and the name of the program is `ZAIF_S###_FLBOOKING_SEL_SCREEN`

1. Create a selection screen.
   Use the following data:

Field	Value
Transaction	SE38
New program	ZAIF_S###_FLBOOKING_SEL_SCREEN

Task 4: Customize the Error-Handling

You want to change the single index table from the standard table to your own. Assign it to your interface in the namespace with specific features. Add your program and screen number for the selection.

To fill the interface specific fields of your index table, the system needs information from which the interface fields are taken. Use the interface specific features to add the necessary Information.

1. Assign the new single index table to your interface.

   Use the following data:

Field	Value
Message Index Table Name	**ZS###_FBOOK_SIDX**
Program Name	**ZAIF_S###_FLBOOKING_SEL_SCREEN**
Screen Number	**0001**

2. Add the fields from your index tables as interface keyfields.

   Use the following data:

Field	Value
Field Sequence Number	**10**
Key Field Name	**AGENCY**
Data Element	**S_AGNCYNUM**
Name Select-Options / Parameter	**S_AGENCY**
Field Is Select-Option	**X**
Field Name	**TRAVEL_AGENCY-AGENCYNUM**
Raw or SAP Structure	**Source Structure**
Multi. Selection Type	**Single selection**
Icon	**@KR@**
Tooltip	**Agency**

Field	Value
Field Sales Agent	
Parent Field Sequence Number	10
Field Sequence Number	**20**
Key Field Name	**SALES_AGENT_ID**

Field	Value
Data Element	**ZAIF_T100_SALES_AGENT_ID**
Name Select-Options / Parameter	**S_AGENT**
Field Is Select-Option	**X**
Field Name	**SALES_AGENT-SALES_AGENT_ID**
Raw or SAP Structure	**Source Structure**
Multi. Selection Type	**Single selection**
Icon	**@VV@**
Tooltip	**Sales Agent**

Field	Value
Field Airport From	
Field Sequence Number	**30**
Key Field Name	**AIRPORT_FROM**
Data Element	**S_FROMAIRP**
Name Select-Options / Parameter	**S_AIRFRO**
Field Is Select-Option	**X**
Field Name	**CUSTOMER_DATA-FLIGHT_BOOKINGS-AIRPORT_FROM**
Raw or SAP Structure	**Source Structure**
Multi. Selection Type	**Multiple selection**
Message Index Table Name	**ZS###_FBOOK_MIDX**

Field	Value
Field Airport To	
Field Sequence Number	**40**
Key Field Name	**AIRPORT_TO**
Data Element	**S_TOAIRP**
Name Select-Options / Parameter	**S_AIRTO**
Field Is Select-Option	**X**
Field Name	**CUSTOMER_DATA-FLIGHT_BOOKINGS-AIRPORT_TO**
Raw or SAP Structure	**Source Structure**
Multi. Selection Type	**Multiple selection**

Field	Value
Message Index Table Name	**ZS###_FBOOK_MIDX**

Field	Value
Field Customer	
Field Sequence Number	**50**
Key Field Name	**CUSTOMER**
Data Element	**S_CUSTOMER**
Name Select-Options / Parameter	**S_CUSTOM**
Field Is Select-Option	**X**
Field Name	**CUSTOMERS-ID**
Raw or SAP Structure	**Dest. Structure**
Multi. Selection Type	**Multiple selection**
Message Index Table Name	**ZS###_FCUST_MIDX**

3. Test your development.

Unit 6
Solution 11

Create and Use Index Tables

Business Scenario

You want to help business users search for specific message content in *Monitoring and Error Handling*. To enable the search, create interface-specific index tables and an interface-specific selection screen. The index tables enable you to store the search fields' values. The interface-specific selection screen displays as an additional sub-selection screen on the selection screen of *Monitoring and Error Handling*.

Note:
This exercise depends on completing the exercise **Set Additional Actions**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create an Interface-Specific Single Index Table

1. Create a single index table.
   Use the following data:

Field	Data Element
Transaction	SE11
Source table	/AIF/STD_IDX_TBL
Target table	ZS###_FBOOK_SIDX
Package	ZAIF_S###
Add the following fields:	
AGENCY	**S_AGNCYNUM**
SALES_AGENT_ID	**ZAIF_T100_SALES_AGENT_ID**

   a) Navigate to transaction *SE11*

   b) Find the standard index table `/AIF/STD_IDX_TBL`.

   c) Choose *Copy*.

   d) In the pop-up, enter the new name **ZS###_FBOOK_SIDX** and the *Package*.

   e) Choose *Execute*.

f) Choose *Change*.

g) Enter the data from the table.

h) Save and activate your table.

Task 2: Create a Multi Index Table

To store values that can occur multiple times, create a multi index table in order. The values are contained in a structure that can exist more than once in the interface. For this interface, *AIRPORT_FROM*, *AIRPORT_TO*, and *CUSTOMERS-ID* can occur multiple times. Searches for those values require a multi index table. The basic version of a multi index table appears like this:

Field	Key Field	Initial Values	Data Element	Group
MANDT	X	X	MANDT	
MSGGUID	X	X	GUID_32	
COUNTER	X	X	INT4	
.INCLUDE			/AIF/IFKEYS	AIFKEYS
.INCLUDE			/AIF/ADMIN	ADMIN
PID			SXMSPID	

The table `/AIF/MIDX_TEMPL` is already a basic version of the multi index table found in the system. The two airport fields are from the source structure and the customerid from the target structure. They cannot be mixed in one index table. The customer ID requires a second index table because it is not related to the airport fields. Create two multi index tables.

1. Create the multi index table for the source structure.

   Use the following data:

Field	Value
Transaction	SE11
Source table	/AIF/MIDX_TEMPL
New table	ZS###_FBOOK_MIDX
Airport Fields	
AIRPORT_FROM	**`S_FROMAIRP`**
AIRPORT_TO	**`S_TOAIRP`**

a) Navigate to the transaction provided in the step.

b) Copy the basic version of the source provided in the step.

c) For a name, enter the value of the new table provided in the step.

d) Open your new table.

e) Switch to change mode.

f) Enter the data provided in the table above, below **Airport Fields** .

g) Save and activate your table.

2. Create the multi index table for customer ID.

   Use the following data:

Field	Value
Transaction	SE11
Source table	/AIF/MIDX_TEMPL
New table	ZS###_FCUST_MIDX
CUSTOMER	**S_CUSTOMER**

   a) Navigate to the transaction provided in the step.

   b) Copy the source provided in the step.

   c) For a name, enter the value of the new table provided in the step.

   d) Open your new table.

   e) Switch to change mode.

   f) Enter the data provided in the table.

   g) Save and activate your table.

Task 3: Create an Interface-Specific Selection Screen

You need an interface-specific screen for selecting from both the single index and multi index tables. You need a module pool and must program the screen. For every field not part of the standard single index table, you need variables and select-options or parameters.

In your case, these are fields for agency, sales agent, two airport fields, and the customerid. The screen must be type subscreen and the name of the program is `ZAIF_S###_FLBOOKING_SEL_SCREEN`

1. Create a selection screen.

   Use the following data:

Field	Value
Transaction	SE38
New program	ZAIF_S###_FLBOOKING_SEL_SCREEN

   a) Using the data provided in the step, call the transaction and create a new program.

   b) Choose *Create* and enter a title.

   c) Select *Type Module Pool*.

   d) Choose *Save*.

   e) Optional: If you are asked for a package and transport request, enter your package and your transport request.

f) Create a selection sub-screen that has your key fields as parameters. Create variables with the data command for each field. Create the screen as shown below.

```
PROGRAM zaif_t100_flbooking_sel_screen.

DATA: lv_agency       TYPE s_agncynum,
      lv_agent        TYPE zaif_t100_sales_agent_id,
      lv_airport_from TYPE s_fromairp,
      lv_airport_to   TYPE s_toairp,
      lv_customer     TYPE s_customer.

SELECTION-SCREEN BEGIN OF SCREEN 0001 AS SUBSCREEN.
SELECT-OPTIONS: s_agency FOR lv_agency,
                s_agent  FOR lv_agent,
                s_airfro FOR lv_airport_from,
                s_airto  FOR lv_airport_to,
                s_custom FOR lv_customer.
SELECTION-SCREEN END OF SCREEN 0001.
```

g) Save and activate your program.

You can set the dictionary flag for the selection texts.

h) Activate the selection texts.

Task 4: Customize the Error-Handling

You want to change the single index table from the standard table to your own. Assign it to your interface in the namespace with specific features. Add your program and screen number for the selection.

To fill the interface specific fields of your index table, the system needs information from which the interface fields are taken. Use the interface specific features to add the necessary Information.

1. Assign the new single index table to your interface.

   Use the following data:

Field	Value
Message Index Table Name	**ZS###_FBOOK_SIDX**
Program Name	**ZAIF_S###_FLBOOKING_SEL_SCREEN**
Screen Number	**0001**

   a) Navigate to transaction `/AIF/CUST` under *Error Handling → Namespace-Specific Features*.

   b) Select your namespace.

   c) Switch to change mode and select your interface.

   If your interface is not listed, create the entry for your interface with the new entries button.

   d) Change the details of your interface.

   e) Enter the data provided in the table.

   f) Save and return to the *AIF Customizing* menu.

2. Add the fields from your index tables as interface keyfields.

Use the following data:

Field	Value
Field Sequence Number	**10**
Key Field Name	**AGENCY**
Data Element	**S_AGNCYNUM**
Name Select-Options / Parameter	**S_AGENCY**
Field Is Select-Option	**X**
Field Name	**TRAVEL_AGENCY-AGENCYNUM**
Raw or SAP Structure	**Source Structure**
Multi. Selection Type	**Single selection**
Icon	**@KR@**
Tooltip	**Agency**

Field	Value
Field Sales Agent	
Parent Field Sequence Number	10
Field Sequence Number	**20**
Key Field Name	**SALES_AGENT_ID**
Data Element	**ZAIF_T100_SALES_AGENT_ID**
Name Select-Options / Parameter	**S_AGENT**
Field Is Select-Option	**X**
Field Name	**SALES_AGENT-SALES_AGENT_ID**
Raw or SAP Structure	**Source Structure**
Multi. Selection Type	**Single selection**
Icon	**@VV@**
Tooltip	**Sales Agent**

Field	Value
Field Airport From	
Field Sequence Number	**30**
Key Field Name	**AIRPORT_FROM**
Data Element	**S_FROMAIRP**
Name Select-Options / Parameter	**S_AIRFRO**
Field Is Select-Option	**X**

Field	Value
Field Name	**CUSTOMER_DATA-FLIGHT_BOOKINGS-AIRPORT_FROM**
Raw or SAP Structure	**Source Structure**
Multi. Selection Type	**Multiple selection**
Message Index Table Name	**ZS###_FBOOK_MIDX**

Field	Value
Field Airport To	
Field Sequence Number	**40**
Key Field Name	**AIRPORT_TO**
Data Element	**S_TOAIRP**
Name Select-Options / Parameter	**S_AIRTO**
Field Is Select-Option	**X**
Field Name	**CUSTOMER_DATA-FLIGHT_BOOKINGS-AIRPORT_TO**
Raw or SAP Structure	**Source Structure**
Multi. Selection Type	**Multiple selection**
Message Index Table Name	**ZS###_FBOOK_MIDX**

Field	Value
Field Customer	
Field Sequence Number	**50**
Key Field Name	**CUSTOMER**
Data Element	**S_CUSTOMER**
Name Select-Options / Parameter	**S_CUSTOM**
Field Is Select-Option	**X**
Field Name	**CUSTOMERS-ID**
Raw or SAP Structure	**Dest. Structure**
Multi. Selection Type	**Multiple selection**
Message Index Table Name	**ZS###_FCUST_MIDX**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Error Handling → Interface-Specific Features*.

b) Select your interface.

c) Select *Define Key Fields for Multi*.

d) Search.

e) Choose *New Entries* and maintain the key fields provided in the Field Agency and Field Sales Agent tables.

f) Using the data provided in the Field Airport From table, create entries for your multi index table.

Note:
To show the corresponding input fields, choose *Enter* after selecting *Multiple Selection*.

g) Using the data provided in the Field Airport To and Field Customer tables, create more entries.

Note:
To enter the correct field name, change the value of *Raw or SAP Structure* to **`Dest. Structure`**.

h) Save your data.

3. Test your development.

a) Execute transaction `/AIF/ERR` and enter your interface's name.

b) Choose *Enter*.

The previously defined selection screen displays as an additional sub screen. In this sub-selection screen, you can select for specific message content.

c) Choose *Execute*.

d) Expand the tree in *Data Content* in *Monitoring and Error Handling*.

The two additional nodes, the agency number and the sales agent number, are additional grouping criteria.

e) To test the behavior of the selection screen, return to the selection screen and enter an agency number.

Note:
Since the single index table was changed, the messages processed previously are no longer displayed in *Monitoring and Error Handling*.

Only the messages containing the agency number display.

Unit 6
Exercise 12

Create Recipients and Alerts

Business Scenario

The SAP Application Interface Framework allows you to notify business users about errors in interfaces they are responsible for. A simple way to do this was already shown at the beginning of the training with recipient ALL_INTERFACES. There are some further options, for example, creating an own alert category or key field dependent notifications.

Note:
This exercise depends on completing the exercise **Create And Use Index Tables**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Simply Assign a Recipient

1. Create a recipient.
   Use the following data:

Field	Value
Define Recipients	IF_FLBOOKING

2. Assign the recipient to an interface.
   Use the following data:
   Interface: **`FLBOOKING`**
3. Assign your user to a recipient.
4. Test your development.
5. Assign the alert category to your namespace.
   Use the following data:

Field	Value
Alert Category	**`ZAIF_TRAINING`**
Alert Category Namespace	**`NAMESPACE`**
Alert Category IF Name	**`INTERFACENAME`**

Field	Value
Alert Category IF Version	**INTERFACEVERSION**

Note:
If you use the standard names for the container elements (NS, IF Name, and IF Version), you need not maintain those fields.

In the alert category, you can maintain a text that gives the user receiving emails information such as the following:

- A hint about how to solve an error
- A link that forwards the user to the system and starts a transaction, for example, *Monitoring and Error Handling*

This category is already in the system and you can reuse this alert category.

Note:
Alert category `/AIF/ALERT_CAT_DEF` is delivered with AIF. If no other alert category is defined, this alert category is used.

Task 2: Create Key Field Dependent Recipients and Alerts

If you use your own single index table, and not the AIF Standard single index table, you can use all of this table's added keyfields to send interfaces to different receivers. Which receivers you send them to depends on the content of these fields.

To create key field dependent alerts, you must create a recipient assignment table. This table must contain the key fields that you use to determine the recipient. AIF delivers a copy template for this table named `/AIF/T_ALRT_DEF`. Copy it to *ZS###_FBOOK_ALRT* and add the sales agency as additional field.

1. Create a recipient assignment table.
   Use the following data:

Field	Value
Source table	/AIF/T_ALRT_DEF
New table name	ZS###_FBOOK_ALRT
AGENCY	**S_AGNCYNUM**

2. Create a recipient for an agency.
   Use the following data:

Field	Value
Agency	AGENCY_120
Second transaction	/AIF/MYRECIPIENTS

3. Configure an alert.

Interface entry: **S###/FLBOOKING/1**

Field	Value
Recipient Assignment Table	**ZS###_FBOOK_ALRT**
Alert Category IF Namespace	**NAMESPACE**
Alert Category IF Name	**INTERFACENAME**
Alert Category IF Version	**INTERFACEVERSION**
All Message Types	Select

4. Create a key field for a recipient.

   Use the following data:

Field	Value
Field Name in Alert Recipient Assignment	**AGENCY**
Category Field Name	**AGENCYNUM**
Relevant for Recipient Determination	Select

5. Assign recipients and key fields.

   Use the following data:

Field	Value
Recipient assignment table	**ZS###_FBOOK_ALRT**
New entries	
NS	**S###**
IFNAME	**FLBOOKING**
IFVERSION	**1**
NSRECIP	**S###**
RECIPIENT	**AGENCY_120**
AGENCY	**120**

6. Test your development.

   If you use the same agency number in your test data as you do in your recipient assignment, you receive an alert. If you use a different number, you do not receive an alert. Because the key field's value is passed to the alert, you also see the agency number in the alert text.

Note:
If you do not receive an alert email, you may need to first confirm an older alert.

Unit 6
Solution 12

Create Recipients and Alerts

Business Scenario

The SAP Application Interface Framework allows you to notify business users about errors in interfaces they are responsible for. A simple way to do this was already shown at the beginning of the training with recipient ALL_INTERFACES. There are some further options, for example, creating an own alert category or key field dependent notifications.

Note:
This exercise depends on completing the exercise **Create And Use Index Tables**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Simply Assign a Recipient

1. Create a recipient.
   Use the following data:

Field	Value
Define Recipients	IF_FLBOOKING

   a) In *Customizing* for the SAP Application Interface Framework, navigate to *Error Handling → Namespace-Specific Features*.

   b) Enter your namespace.

   c) Select *Define Recipients* and choose *New Entries*.

   d) Enter the *Recipient for Alert*value provided in the step and a description.

   e) Save your data.

2. Assign the recipient to an interface.
   Use the following data:

   Interface: **`FLBOOKING`**

   a) In *AIF Customizing*, navigate to *Error Handling → Interface-Specific Features*.

   b) Select the interface provided in the step.
   If your Interface does not exist, create the entry using *new_entries*.

c) Choose *Assign Recipients* without key fields and choose *New Entries*.

d) Enter the recipient you created in the last step.

e) Save.

3. Assign your user to a recipient.

a) Execute transaction `/AIF/MYRECIPIENTS`.

b) Choose *New Entries* and enter the recipient created in a previous step.

c) To see the interface in the interface monitor, select *Overview*.

d) Save.

4. Test your development.

a) Execute transaction `/AIF/IFMON`.

b) Send a new message.
If the message runs into an error, you receive an alert in your alert inbox (transaction `ALRTINBOX`).

c) Start the transaction provided in the step.
The message of your interface displays. If you are still assigned to the `ALL_INTERFACES` recipient, you see both recipients in your inbox.

Note:
If your email address is maintained in the system, you can also receive the alerts as emails.

5. Assign the alert category to your namespace.

Use the following data:

Field	Value
Alert Category	**`ZAIF_TRAINING`**
Alert Category Namespace	**`NAMESPACE`**
Alert Category IF Name	**`INTERFACENAME`**
Alert Category IF Version	**`INTERFACEVERSION`**

Note:
If you use the standard names for the container elements (NS, IF Name, and IF Version), you need not maintain those fields.

In the alert category, you can maintain a text that gives the user receiving emails information such as the following:

- A hint about how to solve an error
- A link that forwards the user to the system and starts a transaction, for example, *Monitoring and Error Handling*

This category is already in the system and you can reuse this alert category.

Note:
Alert category `/AIF/ALERT_CAT_DEF` is delivered with AIF. If no other alert category is defined, this alert category is used.

a) Execute transaction `/AIF/CUST` and choose *Error Handling → Namespace-Specific Features*.

b) Enter your namespace and choose *Configure Alerts*.

c) Choose *New Entries* and enter the interface value provided in a previous step.

d) Enter the data provided in the table.

e) Save the data.

f) Send some test data. The alert email that you receive contains the text of the alert category provided in the step.

Note:
If you do not receive an alert email, you may need to first confirm an older alert.

Task 2: Create Key Field Dependent Recipients and Alerts

If you use your own single index table, and not the AIF Standard single index table, you can use all of this table's added keyfields to send interfaces to different receivers. Which receivers you send them to depends on the content of these fields.

To create key field dependent alerts, you must create a recipient assignment table. This table must contain the key fields that you use to determine the recipient. AIF delivers a copy template for this table named `/AIF/T_ALRT_DEF`. Copy it to *ZS###_FBOOK_ALRT* and add the sales agency as additional field.

1. Create a recipient assignment table.

   Use the following data:

Field	Value
Source table	/AIF/T_ALRT_DEF
New table name	ZS###_FBOOK_ALRT
AGENCY	**S_AGNCYNUM**

a) Execute transaction **SE11**.

b) Copy the source table provided in the step.

c) For your table's name, enter the new table name provided in the step.

d) Enter your recipient assignment table into *Database* table and choose *Change*.

e) To add the field *AGENCY* to your table, enter the data from the AGENCY Field table.

f) Save and activate the table.

2. Create a recipient for an agency.

   Use the following data:

Field	Value
Agency	AGENCY_120
Second transaction	/AIF/MYRECIPIENTS

   a) Execute transaction `/AIF/CUST`.

   b) Navigate to *Error Handling → Namespace-Specific Features* and select *Define Recipients*.

   c) In *New Entries*, enter the *Agency* value provided in the step and a description.

   d) Call the second transaction provided in the step and assign your user to the recipient.

3. Configure an alert.

   Interface entry: **`S###/FLBOOKING/1`**

Field	Value
Recipient Assignment Table	**`ZS###_FBOOK_ALRT`**
Alert Category IF Namespace	**`NAMESPACE`**
Alert Category IF Name	**`INTERFACENAME`**
Alert Category IF Version	**`INTERFACEVERSION`**
All Message Types	Select

   a) In AIF Customizing (transaction `/AIF/CUST`), navigate to *Error Handling → Namespace-Specific Features*.

   b) Select *Configure Alerts*.

   c) Select the value of your interface entry provided in the step.

   d) For the interface entry, enter the values provided in the table.

   e) Save.

4. Create a key field for a recipient.

   Use the following data:

Field	Value
Field Name in Alert Recipient Assignment	**`AGENCY`**
Category Field Name	**`AGENCYNUM`**
Relevant for Recipient Determination	Select

   a) In AIF Customizing, navigate to *Error Handling → Interface-Specific Features*.

   b) Select *Define Key Fields for Multi. Search*.

c) Select the key field for your agency.

d) In *Single Selection* sub-screen, enter the data provided in the table.

e) Save.

5. Assign recipients and key fields.

Use the following data:

Field	Value
Recipient assignment table	**ZS###_FBOOK_ALRT**
New entries	
NS	**S###**
IFNAME	**FLBOOKING**
IFVERSION	**1**
NSRECIP	**S###**
RECIPIENT	**AGENCY_120**
AGENCY	**120**

a) Execute transaction **SE16**.

b) Enter the recipient assignment table as given in the table above.

c) Choose *Create Entries*.

d) Enter the data provided in the *Alert* table.

e) Save.

6. Test your development.

If you use the same agency number in your test data as you do in your recipient assignment, you receive an alert. If you use a different number, you do not receive an alert. Because the key field's value is passed to the alert, you also see the agency number in the alert text.

Note:
If you do not receive an alert email, you may need to first confirm an older alert.

a) Navigate to the AIF interface test tool (transaction `/AIF/IFTEST`) and open your test file.

Note:
To produce an alert, use an invalid class. Use the agency that you filled in the assignment table for your recipient.

b) Save your file.

c) Process the data in the XML runtime.

d) Navigate to the Interface Monitor and access *Monitoring and Error Handling*.

e) Select the newest message.
In *Log Messages*, you see an error that the flight could not be created because of an incorrect flight class.

f) Confirm the raised alert.

g) Navigate to the AIF interface test tool (transaction `/AIF/IFTEST`) and open your test file.

Note:
To produce an alert, use an invalid class. Use an agency not assigned to a receiver.

h) Process the data in the XML runtime.

i) Navigate to the Interface Monitor and access *Monitoring and Error Handling*.
You do not see a new entry because you are not assigned to this agency.

j) Navigate to the AIF error handling transaction `/AIF/ERR`.

k) Search for your previously used agency.
Here you find the created message. All messages are shown, not only those you have been assigned to.

Unit 6
Exercise 13

Hide Fields And Structures In Monitoring

Business Scenario

You want to prevent every user being able to see the sales agent data of the message. Hide the corresponding structure. You also want to prevent a business user seeing the credit card field of the customer details. Hide the corresponding field.

> Note:
> This exercise depends on completing the exercise **Create Recipients And Notifications**.

> Note:
> In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Hide Fields And Structures In Monitoring

1. Hide structures.

   Use the following data:

   Field structure: **`SALES_AGENT`**

   > Note:
   > The changes only take effect after you restart the *Monitoring and Error Handling* transaction.

2. Hide fields.

   Use the following data:

Field	Value
Index	`10`
Field Path	`CUSTOMER_DATA-CUSTOMER_DETAILS-CREDITCARD`

   > Note:
   > The changes only take effect after you restart the *Monitoring and Error Handling* transaction.

Unit 6
Solution 13

Hide Fields And Structures In Monitoring

Business Scenario

You want to prevent every user being able to see the sales agent data of the message. Hide the corresponding structure. You also want to prevent a business user seeing the credit card field of the customer details. Hide the corresponding field.

> Note:
> This exercise depends on completing the exercise **Create Recipients And Notifications**.

> Note:
> In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Hide Fields And Structures In Monitoring

1. Hide structures.

   Use the following data:

   Field structure: **`SALES_AGENT`**

   > Note:
   > The changes only take effect after you restart the *Monitoring and Error Handling* transaction.

   a) Navigate to AIF Customizing (transaction `/AIF/CUST`) and navigate to *Error Handling → Define Interface-Specific Features*.

   b) Select your namespace, interface, and version.

   c) Choose *Hide Structures* and create a new entry.

   d) Using the F4 help, select the value for the field structure provided in the step.

   e) Save.

   f) In *Monitoring and Error Handling* (transaction `/AIF/ERR`), check the result. You no longer see the *SALES_AGENT* structure in *Data Structure*.

2. Hide fields.

   Use the following data:

Field	Value
Index	**10**
Field Path	**CUSTOMER_DATA-CUSTOMER_DETAILS-CREDITCARD**

Note:
The changes only take effect after you restart the *Monitoring and Error Handling* transaction.

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) and navigate to *Error Handling → Define Interface-Specific Features*.

b) Select your namespace, interface, and version.

c) Choose *Hide Fields of Structure* and create a new entry.

d) Enter the data provided in the step.

e) In *Monitoring and Error Handling* (transaction `/AIF/ERR`), check the result.

Unit 7
Exercise 14

Send An Interface to the XML Enabler

Business Scenario

In previous exercises, you built an interface to process customer and flight bookings. You can trigger it with the AIF test tool. Live, you no longer want to create and send a message to AIF with the test tool. Instead, you want to use a report that sends interfaces to the XML Runtime with the method `/AIF/CL_ENABLER_XML=>TRANSFER_TO_AIF` .You can use the copy template *ZAIF_BIT750_FLBOOKING*.

Note:
This exercise depends on completing the exercise **Hide Fields And Structures In Monitoring**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Send An Interface To The XML Enabler And Check The Result

1. Edit the report.
   Use the following data:

Field	Value
Report	ZAIF_BIT750_FLBOOKING
Target	ZAIF_S###_FLBOOKING
LS_RAW_STRUCT	ZAIF_S###_RAW_FLIGHT
Method	
Class	`/AIF/CL_ENABLER_XML`
Method	`TRANSFER_TO_AIF`
IS_ANY_STRUCTURE	LS_RAW_STRUCT

2. Execute the report.
3. Check the result.

Unit 7
Solution 14

Send An Interface to the XML Enabler

Business Scenario

In previous exercises, you built an interface to process customer and flight bookings. You can trigger it with the AIF test tool. Live, you no longer want to create and send a message to AIF with the test tool. Instead, you want to use a report that sends interfaces to the XML Runtime with the method `/AIF/CL_ENABLER_XML=>TRANSFER_TO_AIF` .You can use the copy template *ZAIF_BIT750_FLBOOKING*.

Note:
This exercise depends on completing the exercise **Hide Fields And Structures In Monitoring**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Send An Interface To The XML Enabler And Check The Result

1. Edit the report.

   Use the following data:

Field	Value
Report	ZAIF_BIT750_FLBOOKING
Target	ZAIF_S###_FLBOOKING
LS_RAW_STRUCT	ZAIF_S###_RAW_FLIGHT
Method	
Class	`/AIF/CL_ENABLER_XML`
Method	`TRANSFER_TO_AIF`
IS_ANY_STRUCTURE	LS_RAW_STRUCT

   a) Navigate to *SE38* and enter the report provided in the step.

   b) Choose *Copy*.

   c) Change the name of the target program to the target provided in the step.

   d) Open the report and replace the defined data type for *LS_RAW_STRUCT* with the value provided in the step.

e) At the end of the report, call the method described in the Method table.

f) For exporting parameter, enter the value for *IS_ANY_STRUCTURE* provided in the step.
The method call resembles the following code:

```
CALL METHOD /aif/cl_enabler_xml=>transfer_to_aif
  EXPORTING
    is_any_structure = ls_raw_struct
```

g) Save and activate the report.

2. Execute the report.
   a) To execute the report, press F8 and create a new message.
   b) Enter an agency and customer information.
3. Check the result.
   a) To check the result, view it in AIF Monitoring and Error Handling (transaction `/AIF/ERR`).
   You see a new message in your *FLBOOKING* interface. Its success depends on the data you sent.

Unit 8
Exercise 15

Generate the Structures for an Existing IDoc and Monitor it - IDoc Scenario 1

Business Scenario

You can monitor IDocs with AIF. With IDoc Scenario 1, you can monitor IDocs processed by the ALE runtime without AIF.

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Generate an IDoc Structure

1. Generate a DDIC structure.

   Use the following data:

Field	Value
General Details area	
Basic Type	**ZAIF_S###_FLBOOK_CREATE01**
Message Type	**FLIGHTBOOKING_CREATEFROMDAT**
Extension	<remains empty>
Root Structure Name	ZAIF_S###_FLBOOK_CREATE01
Destination for Remote Search	<remains empty>
Create or update Structure	<flag>
Create Interface	<flag>, when flagged, hit *Enter* on your keyboard
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**

Field	Value
Interface Name	**IDOC_SCEN1**
Interface Version	**1**
Interface Description	**Interface for IDoc Scenario 1**
Interface Direction	Both
IDoc Processing Scenario	**01 (Standard IDoc Runtime)**
Customizing Request	<the one given to you from the instructor>

2. Check the created structure.
   Compare to the following data:
   Interface: **IDOC_SCEN1**

Component	Type
EDIDC	**STRUC**
E1SBO_CRE	**STRUC**
.INCLUDE(E1SBO_CRE)	
E1BPSBONEW	**STRUC**
E1BPPAREX	**TAB**

3. Check the created interface.
   Use the following data:

Field	Value
Namespace	**S###**
Interface Name	**IDOC_SCEN1**
Interface Version	**1**
SAP Data Structure	**ZAIF_S###_FLCUST_CREATE01**
Raw Data Structure	**ZAIF_S###_FLCUST_CREATE01**
Move Corresponding Structures	**X**

Field	Value
Application Engine ID	IDoc
Persistency Engine ID	IDoc
Selection Engine ID	IDoc Control Records
Logging Engine ID	IDoc Status Records

4. Check additional interface settings.
   Compare to the following data:

Field	Value
Message type	**FLIGHTBOOKING_CREATEFROMDAT**
Basic type	**ZAIF_S###_FLBOOK_CREATE01**

5. Optional: Maintain the recipient.

Note:
If your user is still assigned to the *ALL_INTERFACES* recipient, you can skip this step.

Task 2: Test the Interface

You want to test the interface. You can do this through creating an IDoc with test transaction `WE19`.

1. Test a successful IDoc.

   Use the following data:

Field	Value
Basic Type	ZAIF_S###_FLBOOK_CREATE01
Port	**SAP<SYS-ID> (replace SYS-ID with your system ID) e.g. SAPT41**
Partner No. Sender	**ZAIF_S###**
Part. Type Sender	**LS**
Partner No. Receiver	**ZAIF_S###**
Part. Type Receiver	**LS**
Message Type	**FLIGHTBOOKING_CREATEFROMDAT**

2. Create a flight booking.

   Use the following data:

Field	Value
Airline ID	**LH**
Connection ID	**0400**
Flight Date	**20140412**
Customer ID	**7**
Class	**Y**
Agency	**109**

Note:
It is possible the data in Flight Details does not work because, for example, the flight is full. In that case, choose a flight from table *SFLIGHTS* with available seats. Change your test data accordingly.

Field	Value
Namespace	**S###**
Interface	**IDOC_SCEN1**
Interface Version	**1**
Generic Selection	**Today**
Status Selection	*Select All*

3. Test an IDoc with an error.
   Use the following data:

   Flight class: **Z**

4. Cancel the error in monitoring and error handling.
   Use the following data:

Field	Value
Namespace	**S###**
Interface	**IDOC_SCEN1**
Interface Version	**1**
Generic Selection	**Today**
Status Selection	*Select Application Errors and Technical Errors*

Task 3: Solve the Errors in Monitoring and Error Handling
You want to repair and restart the second faulty IDoc. Before you can solve this error in Monitoring and Error Handling, you must define the corresponding field as editable.

1. Define the field as editable.
   Use the following data:

   Namespace: **S###**

Field	Value
Index	**10**
Field Path	**E1SBO_CRE-E1BPSBONEW-CLASS**

2. Solve the error.
   Use the following data:

Field	Value
Namespace	**S###**
Interface	**IDOC_SCEN1**
Interface Version	**1**
Generic Selection	**Today**
Status Selection	*Select Application Errors and Technical Errors*

Allowed flight class: **Y**

Unit 8
Solution 15

Generate the Structures for an Existing IDoc and Monitor it - IDoc Scenario 1

Business Scenario

You can monitor IDocs with AIF. With IDoc Scenario 1, you can monitor IDocs processed by the ALE runtime without AIF.

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Generate an IDoc Structure

1. Generate a DDIC structure.

   Use the following data:

Field	Value
General Details area	
Basic Type	**ZAIF_S###_FLBOOK_CREATE01**
Message Type	**FLIGHTBOOKING_CREATEFROMDAT**
Extension	<remains empty>
Root Structure Name	ZAIF_S###_FLBOOK_CREATE01
Destination for Remote Search	<remains empty>
Create or update Structure	<flag>
Create Interface	<flag>, when flagged, hit *Enter* on your keyboard
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**

Field	Value
Interface Name	**IDOC_SCEN1**
Interface Version	**1**
Interface Description	**Interface for IDoc Scenario 1**
Interface Direction	Both
IDoc Processing Scenario	**01 (Standard IDoc Runtime)**
Customizing Request	<the one given to you from the instructor>

a) Navigate to the *Generate IDoc Structure and Interface Definition* report (transaction `/AIF/IDOC_GEN`).

b) To create the corresponding DDIC structure, use the data in the table above.

c) Use the F4–Help to enter your transport request in the *Workbench Request/Task* field.

d) Choose *Execute*.

2. Check the created structure.
   Compare to the following data:

   Interface: **IDOC_SCEN1**

Component	Type
EDIDC	**STRUC**
E1SBO_CRE	**STRUC**
.INCLUDE(E1SBO_CRE)	
E1BPSBONEW	**STRUC**
E1BPPAREX	**TAB**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Interfaces*.

b) Select your namespace and open the interface provided in the table.

c) Double click on the raw data structure.
A new window opens and the structure displays (transaction `SE11` is started).

d) Check that there is the structure and substructure as provided in the Structure table.

Note:
Instead of using the forward navigation, you can also access transaction `SE11` directly to display your generated structure.

e) Execute transaction `SE80` and check if the generated structures are assigned to your package.

Note:
If you cannot see the structure and table types in your package, choose the package's context menu. Select *Other Functions → Rebuild Object List*.

3. Check the created interface.

   Use the following data:

Field	Value
Namespace	**S###**
Interface Name	**IDOC_SCEN1**
Interface Version	**1**
SAP Data Structure	**ZAIF_S###_FLCUST_CREATE01**
Raw Data Structure	**ZAIF_S###_FLCUST_CREATE01**
Move Corresponding Structures	**X**

Field	Value
Application Engine ID	IDoc
Persistency Engine ID	IDoc
Selection Engine ID	IDoc Control Records
Logging Engine ID	IDoc Status Records

   a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Interfaces*.

   b) Enter your namespace.

   c) To check that the interface created by the report exists, display the interface details. Check that the details are the same as provided in the Interface Details table.

   d) Return to AIF Customizing (transaction `/AIF/CUST`) and select *Interface Development → Additional Interface Properties → Specific Interface Engines*.

   e) Enter your namespace and select your interface **IDOC_SCEN1**.

   f) Maintain the engines provided in the Engines table.

4. Check additional interface settings.

   Compare to the following data:

Field	Value
Message type	**FLIGHTBOOKING_CREATEFROMDAT**
Basic type	**ZAIF_S###_FLBOOK_CREATE01**

a) Navigate to the AIF Customizing (transaction `/AIF/CUST`) and select *Interface Development → Additional Interface Properties → Assign IDOC Types*.

b) Maintain the message types provided in the table.

5. Optional: Maintain the recipient.

Note:
If your user is still assigned to the *ALL_INTERFACES* recipient, you can skip this step.

a) Navigate to *System Configuration → Assign Recipients*.

b) Assign your user to the *ALL_INTERFACES* recipient.

Task 2: Test the Interface

You want to test the interface. You can do this through creating an IDoc with test transaction `WE19`.

1. Test a successful IDoc.

   Use the following data:

Field	Value
Basic Type	ZAIF_S###_FLBOOK_CREATE01
Port	**SAP<SYS-ID> (replace SYS-ID with your system ID) e.g. SAPT41**
Partner No. Sender	**ZAIF_S###**
Part. Type Sender	**LS**
Partner No. Receiver	**ZAIF_S###**
Part. Type Receiver	**LS**
Message Type	**FLIGHTBOOKING_CREATEFROMDAT**

   a) Execute transaction: `WE19`.

   b) Select *Basic Type* and enter the provided value for the corresponding field.

   c) Execute.
   An empty IDoc structure displays.

   d) Double click on *EDIDC*.

   e) Using the data provided in the table, maintain the data for *Receiver and Sender* and *Message Type*.

   f) Close the dialog.

2. Create a flight booking.

   Use the following data:

Field	Value
Airline ID	**LH**

Field	Value
Connection ID	**0400**
Flight Date	**20140412**
Customer ID	**7**
Class	**Y**
Agency	**109**

Note:
It is possible the data in Flight Details does not work because, for example, the flight is full. In that case, choose a flight from table *SFLIGHTS* with available seats. Change your test data accordingly.

Field	Value
Namespace	**S###**
Interface	**IDOC_SCEN1**
Interface Version	**1**
Generic Selection	**Today**
Status Selection	*Select All*

a) Expand *E1SBO_CRE* and double click *E1BPSBONEW*.
A dialog appears.

b) In transaction *SE16*, select a valid flight from table *SFLIGHTS*.

c) Fill the details for a flight with the data provided in the table above.

d) Close the dialog.

e) From the application toolbar, select *Standard inbound*.
An IDoc is created and transfers to the application.

f) Note the IDoc number that displays.

g) Navigate to *Monitoring and Error Handling* (transaction `/AIF/ERR`).

h) Enter the data from the Monitoring and Error Handling table.

Note:
Unfortunately, the interface randomly raises an error. Ask your trainer to fix this error.

i) Execute.

j) In the *Data Messages* view, expand the nodes of your interface *IDOC_SCEN1*.

One IDoc message displays. If the booking is successful, it shows a successful status.

k) Double click the IDoc.
In the *Data Structure* view, the structure of the IDoc displays. In the *Log Messages* view, the last status records of the IDoc display. If the flight booking is created successfully, the following message displays: **`Flight Booking XXX has been created. External reference: XXXXXX`**.

l) In the Data Structure view, double click on one of the structures, for example, *E1BPSBONEW*.
In the *Data Content* view, the data displays.

3. Test an IDoc with an error.
Use the following data:

Flight class: **`Z`**

a) Navigate to transaction *WE19*.

b) Enter the number of the IDoc created before in the *Existing IDoc* field.

c) Execute.
The IDoc created before is the template for creating a new IDoc.

d) To provoke an error in the IDoc processing, change the class to the flight class provided in the step.

e) To process the IDoc, select *Standard Inbound*.

f) To process a second IDoc, choose *Standard Inbound* again.

4. Cancel the error in monitoring and error handling.
Use the following data:

Field	Value
Namespace	**`S###`**
Interface	**`IDOC_SCEN1`**
Interface Version	**`1`**
Generic Selection	**`Today`**
Status Selection	*Select Application Errors and Technical Errors*

a) Navigate to Monitoring and Error Handling (transaction `/AIF/ERR`)

b) Enter the data from the Error Cancellation table.

c) Execute.
You see two IDoc messages in the *Data Messages* view.

d) Select one of the messages.
The *Log Messages* view displays the status records.

e) Double click on the log message **`Flight class can only be Y, C or F`**.
You are forwarded to the erroneous field.

f) Double click on the field.
An error message displays that you cannot edit this field. The error cannot be solved.

g) Cancel the IDoc message.

Task 3: Solve the Errors in Monitoring and Error Handling

You want to repair and restart the second faulty IDoc. Before you can solve this error in Monitoring and Error Handling, you must define the corresponding field as editable.

1. Define the field as editable.

   Use the following data:

   Namespace: **S###**

Field	Value
Index	**10**
Field Path	**E1SBO_CRE-E1BPSBONEW-CLASS**

   a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Error Handling → Namespace-Specific Features*.

   b) Enter the namespace provided in the step.

   c) Create a new entry for the IDoc interface in define namespace specific features.

   d) Return to the AIF Customizing (transaction `/AIF/CUST`) – *Error Handling → Interface-Specific Features*.

   e) Enter your namespace, interface, and version.

   f) Choose *Define Changeable Fields*.

   g) Choose *New Entries* and create a new entry for field *CLASS*.

   h) Enter the data from the CLASS Details table.

   i) Save.
   Now you can solve the error in Monitoring and Error Handling.

2. Solve the error.

   Use the following data:

Field	Value
Namespace	**S###**
Interface	**IDOC_SCEN1**
Interface Version	**1**
Generic Selection	**Today**
Status Selection	*Select Application Errors and Technical Errors*

   Allowed flight class: **Y**

a) In transaction `/AIF/ERR`, enter the data from the Error Cancellation table.

b) Execute.
In the Data Messages view, at least one IDoc message displays.

c) Select the IDoc.

d) In the Log Messages view, double click message **`Flight class can only be Y, C or F`**.
You are forwarded to the erroneous field.

e) Double click on the field.
A dialog appears.

f) Change the value to the allowed flight class provided in the step.

g) Choose *OK*.

h) In *Data Content*, choose *Save*.
The IDoc icon in *Data Messages* changes to *IDoc was edited*.

i) In *Data Messages*, select *Restart*.

j) To check if the restart succeeds, ensure that the message is selected and choose *Read*.
The icon changes to application document posted. In *Log Messages*, a message displays that the flight booking is created.

Unit 8
Exercise 16

Process an IDoc by AIF with IDoc Function Call in Action - IDoc Scenario 2

Business Scenario

In this exercise, you generate the structure for an IDoc and create an interface that can be used to monitor IDocs that are processed via AIF. The interface will call the standard IDoc function inside of an AIF action.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Prepare The Scenario

This exercise requires some basic IDoc configurations. If you already have an existing IDoc, to change the processing from standard processing to AIF processing, adapt the following steps.

These steps process the IDoc with AIF instead of using IDoc standard functionality. These steps do not cover an IDoc's complete setup. They do not include creating a basic type and assignment of a message type. These tasks are not part of AIF.

To process an IDoc through AIF, assign one of the AIF function modules to the corresponding message type-IDoc Basis Type combination as an allowed function.

`/AIF/IDOC_INB_PROCESS_FUNC_0` handles one single IDoc over to AIF and can be used for Scenario 2.

1. Assign a function module to a message type and basic type.
   Use the following data:

Field	Value
Function module	/AIF/IDOC_INB_PROCESS_FUNC_0
Message type	FLCUSTOMER_CREATEFROMDATA
Basic IDoc type	ZAIF_S###_FLCUST_CREATE01

Note:
The transaction to assign a function module to a message type is `WE57`.

2. The function module `/AIF/IDOC_INB_PROCESS_FUNC_0` can now be used for your IDoc. Create the needed processing code for the inbound processing with AIF.
   Use the following data:

Field	Value
Process code	ZAIF_S###_FLCUST_CREATE_01
Function Module	/AIF/IDOC_INB_PROCESS_FUNC_0

Note:
The transaction for IDoc inbound processing codes is `WE42`.

3. Maintain a partner profile.
   Use the following data:

Field	Value
Partner	LS
Logical system	ZAIF_S###
Message type	FLCUSTOMER_CREATEFROMDATA
Process code	**ZAIF_S###_FLCUST_CREATE_01**
Cancel Processing After Syntax Error	Select

Note:
Partner profiles are maintained in transaction `WE20`.

Task 2: Generate Structure and Create Interface

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc. Do this through the AIF IDoc generation report (transaction `/AIF/IDOC_GEN`). The report can also create the interface.

1. Create a DDIC structure out of the basic type of the IDoc.
   Use the following data:

Field	Value
General Details area	
Basic Type	**ZAIF_S###_FLCUST_CREATE01**
Message Type	**FLCUSTOMER_CREATEFROMDATA**
Structure Prefix	**ZAIF_S###_**
Root Structure Name	**ZAIF_S###_FLCUST_CREATE01**
Destination for Remote Search	<remains empty>
Create or update Structure	<flag>
Create Interface	<flag>, when flagged hit *Enter* on your keyboard

Field	Value
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**
Interface Name	**IDOC_SCEN2**
Interface Version	**1**
IDoc Processing Scenario	**02** (AIF Runtime; Call IDoc function in Action)
Interface Description	**Interface for IDoc scenario 2**

2. Check in the AIF Customizing the structures created from the IDoc Generator.

Note:
The structures can also be found in your package ZAIF_S###.

Compare to the following data:

Component	Type
EDIDC	STRUC
E1SCU_CRE	STRUC
.INCLUDE(E1SCU_CRE)	
E1BPSCUNEW	STRUC
E1BPPAREX	TAB

3. Check the interface created from the IDoc generation report. Check the engines chosen from the report. If you find a different result, change the settings to the settings in the solution of this exercise.

Use the following data:

Field	Value
Interface Details	
Namespace	**S###**
Interface Name	**IDOC_SCEN2**
Interface Version	**1**
SAP Data Structure	**ZAIF_S###_FLCUST_CREATE01**
Raw Data Structure	**ZAIF_S###_FLCUST_CREATE01**

Field	Value
Move Corresponding Structures	X

Field	Value
Engine Details	
Application Engine ID	**IDoc**
Persistency Engine ID	**IDoc**
Selection Engine ID	**AIF Index Table**
Logging Engine ID	**AIF Application Log**

4. Create an action inside AIF to post, after all AIF mapping and checking is done, the IDoc with the standard IDoc function module. Define the action AC_IDOC_FLCUST_CREA in the AIF Customizing.

Note:
From the IDoc perspective, the work is done by heading the IDoc to AIF. The functionality for the posting of the data must be done in an Action in AIF.

Use the following data:

Field	Value
Action	**AC_IDOC_FLCUST_CREA**
Action description	**Create customer for flights (S###)**
Commit Mode	*COMMIT WORK*
Commit Level	*After Each Function*
Main Component Type	**ZAIF_S###_FLCUST_CREATE01**
Function number	10
Function module name	ZAIF_S###_FLCUST_CREATE_IDOC
Function group	ZAIF_S###

5. Implement your function module.
Use the following data:

Field	Value
Action number	**10**
Action	AC_IDOC_FLCUST_CREA

6. Define the customer type as changeable.

Note:
If the IDoc contains an incorrect value for customer type, solve errors in Monitoring and Error Handling.

Use the following data:

Field	Value
Namespace	S###
Field path	E1SCU_CRE-E1BPSCUNEW-CUSTTYPE

7. To test the interface, create an IDoc.

Note:
Create the IDoc with the test transaction `WE19`.

Use the following data:

Field	Value
Basic Type	ZAIF_S###_FLCUST_CREATE01
EDIDC Details	
Port	**SAP<SYS-ID>** (replace SYS-ID with your system ID, for example, SAPT41)
Partner No. Sender	**ZAIF_S###**
Part. Type Sender	**LS**
Partner No. Receiver	**ZAIF_S###**
Part. TypeReceiver	**LS**
Message Type	**FLCUSTOMER_CREATEFROMDATA**
New Customer	
Custname	**Friedolin Hirsch**
Street	**Bachstelzenweg 1**
Postcode	**21614**
City	**Buxtehude**
Custtype	<empty>

8. Check the error messages.
9. Solve the error.

Unit 8
Solution 16

Process an IDoc by AIF with IDoc Function Call in Action - IDoc Scenario 2

Business Scenario

In this exercise, you generate the structure for an IDoc and create an interface that can be used to monitor IDocs that are processed via AIF. The interface will call the standard IDoc function inside of an AIF action.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Prepare The Scenario

This exercise requires some basic IDoc configurations. If you already have an existing IDoc, to change the processing from standard processing to AIF processing, adapt the following steps.

These steps process the IDoc with AIF instead of using IDoc standard functionality. These steps do not cover an IDoc's complete setup. They do not include creating a basic type and assignment of a message type. These tasks are not part of AIF.

To process an IDoc through AIF, assign one of the AIF function modules to the corresponding message type-IDoc Basis Type combination as an allowed function.

`/AIF/IDOC_INB_PROCESS_FUNC_0` handles one single IDoc over to AIF and can be used for Scenario 2.

1. Assign a function module to a message type and basic type.

   Use the following data:

Field	Value
Function module	/AIF/IDOC_INB_PROCESS_FUNC_0
Message type	FLCUSTOMER_CREATEFROMDATA
Basic IDoc type	ZAIF_S###_FLCUST_CREATE01

Note:
The transaction to assign a function module to a message type is `WE57`.

   a) Navigate to transaction `WE57` and switch to change mode.

   b) Choose *New Entries*.

c) Enter function module **`/AIF/IDOC_INB_PROCESS_FUNC_0`**.

d) Choose *Function Module* as the function type.

e) Enter the values of the message type and basic IDoc type provided in the step.

f) Choose the direction *Inbound*.

g) Save.
If prompted, enter your transport request.

2. The function module `/AIF/IDOC_INB_PROCESS_FUNC_0` can now be used for your IDoc. Create the needed processing code for the inbound processing with AIF.
Use the following data:

Field	Value
Process code	ZAIF_S###_FLCUST_CREATE_01
Function Module	/AIF/IDOC_INB_PROCESS_FUNC_0

Note:
The transaction for IDoc inbound processing codes is `WE42`.

a) Navigate to transaction `WE42` and switch to change mode.

b) Create a new entry.

c) Enter the value provided in the step for *Process code*.

d) Choose *Processing with ALE service* and *Processing by function module*.

e) Choose *Enter* and save.
A new window opens.

f) Choose the value of *Function Module* provided in the step.

g) Save.

3. Maintain a partner profile.
Use the following data:

Field	Value
Partner	LS
Logical system	ZAIF_S###
Message type	FLCUSTOMER_CREATEFROMDATA
Process code	**`ZAIF_S###_FLCUST_CREATE_01`**
Cancel Processing After Syntax Error	Select

Note:
Partner profiles are maintained in transaction `WE20`.

a) Navigate to `WE20` and open the node *Logical Systems*.

b) Choose the logical system provided in the step.

c) Choose *Create inbound parameters*.

d) Enter the message type provided in the step.

e) In *Inbound Options*, enter the data from the Inbound table.

f) Choose *Trigger Immediately*.

g) Save.

Task 2: Generate Structure and Create Interface

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc. Do this through the AIF IDoc generation report (transaction `/AIF/IDOC_GEN`). The report can also create the interface.

1. Create a DDIC structure out of the basic type of the IDoc.

   Use the following data:

Field	Value
General Details area	
Basic Type	**ZAIF_S###_FLCUST_CREATE01**
Message Type	**FLCUSTOMER_CREATEFROMDATA**
Structure Prefix	**ZAIF_S###_**
Root Structure Name	**ZAIF_S###_FLCUST_CREATE01**
Destination for Remote Search	<remains empty>
Create or update Structure	<flag>
Create Interface	<flag>, when flagged hit *Enter* on your keyboard
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**
Interface Name	**IDOC_SCEN2**
Interface Version	**1**

Field	Value
IDoc Processing Scenario	**02** (AIF Runtime; Call IDoc function in Action)
Interface Description	**Interface for IDoc scenario 2**

a) Navigate to transaction `/AIF/IDOC_GEN`.

b) Enter the data from the IDoc Structure table.

c) Enter your transport request in the *Workbench Request/Task* field.

d) Choose *Execute*.
A structure for your basic type is generated. An interface in your namespace is also created.

2. Check in the AIF Customizing the structures created from the IDoc Generator.

Note:
The structures can also be found in your package ZAIF_S###.

Compare to the following data:

Component	Type
EDIDC	STRUC
E1SCU_CRE	STRUC
.INCLUDE(E1SCU_CRE)	
E1BPSCUNEW	STRUC
E1BPPAREX	TAB

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Interfaces*.

b) Choose your namespace and open your interface (*IDOC_SCEN2*).

c) Double-click on the raw data structure.
A new window opens and the structure displays (transaction `SE11` is started).

d) Using the data in the Structure table, check the structure and substructure.

Note:
Instead of using the forward navigation, you can also access transaction `SE11` directly to display your generated structure.

e) Execute transaction `SE80` and check if the generated structures are assigned to your package.

Note:
If you cannot see the structure and table types in your package, right-click the package. Choose *Other Functions* → *Rebuild Object List*.

3. Check the interface created from the IDoc generation report. Check the engines chosen from the report. If you find a different result, change the settings to the settings in the solution of this exercise.

   Use the following data:

Field	Value
Interface Details	
Namespace	**S###**
Interface Name	**IDOC_SCEN2**
Interface Version	**1**
SAP Data Structure	**ZAIF_S###_FLCUST_CREATE01**
Raw Data Structure	**ZAIF_S###_FLCUST_CREATE01**
Move Corresponding Structures	X

Field	Value
Engine Details	
Application Engine ID	**IDoc**
Persistency Engine ID	**IDoc**
Selection Engine ID	**AIF Index Table**
Logging Engine ID	**AIF Application Log**

   a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development* → *Define Interfaces*.

   b) Enter your namespace.

   c) Check that the interface created by the report exists.

   d) Check that the interface details correspond to the data from the Interface Details table.

   e) Return to AIF Customizing (transaction `/AIF/CUST`) and choose *Interface Development* → *Additional Interface Properties* → *Specific Interface Engines*.

   f) Enter your namespace and choose your interface *IDOC_SCEN2*.

   g) Check the interface details correspond to the data from the Engine Details table.

4. Create an action inside AIF to post, after all AIF mapping and checking is done, the IDoc with the standard IDoc function module. Define the action AC_IDOC_FLCUST_CREA in the AIF Customizing.

Note:
From the IDoc perspective, the work is done by heading the IDoc to AIF. The functionality for the posting of the data must be done in an Action in AIF.

Use the following data:

Field	Value
Action	**AC_IDOC_FLCUST_CREA**
Action description	**Create customer for flights (S###)**
Commit Mode	*COMMIT WORK*
Commit Level	*After Each Function*
Main Component Type	**ZAIF_S###_FLCUST_CREATE01**
Function number	10
Function module name	ZAIF_S###_FLCUST_CREATE_IDOC
Function group	ZAIF_S###

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Actions*.

b) Enter your namespace.

c) Create a new entry.

d) Maintain the data provided in the Action Data table.

e) Navigate to *Define Functions* on the left-hand-side menu and create a new entry.

f) Enter the values of the function number and function module name provided in the step.

g) Choose *Enter*.
A dialog box appears asking if you would like to create the function module.

h) Choose *Yes*.

i) Enter the function group provided in the step.

j) Save.

k) Double click the function module.
Transaction `SE37` starts in a new window. In this function module, the standard IDoc process function IDOC_INPUT_FLCUSTOMER_CREATEFR has to be called. Before the function module is called, the data needs to be transformed from SAP structure to IDoc structure. After calling the IDoc, the status record is added to the application log.

5. Implement your function module.
Use the following data:

Field	Value
Action number	10
Action	AC_IDOC_FLCUST_CREA

a) Call the */AIF/IDOC_ACTION_FUNCTION* function module.
Ensure you pass the correct name of the standard IDoc process function to parameter IV_IDOC_FUNCTION. Call the standard IDoc process function IDOC_INPUT_FLCUSTOMER_CREATEFR.

```
CALL FUNCTION '/AIF/IDOC_ACTION_FUNCTION'
  EXPORTING
    iv_idoc_function = 'IDOC_INPUT_FLCUSTOMER_CREATEFR'
  TABLES
    return_tab       = return_tab
  CHANGING
    data             = data.
```

b) Save and activate your function module.

c) Close transaction `SE37`.

d) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Structure Mapping*.

e) Enter your namespace, interface name, and version.

f) Choose *Assign Actions* and create a new entry.

g) Enter the data from the Assign Actions table.

h) Save.
During the processing of the IDoc, AIF is called and index table and message index table entries are written. Therefore, it is possible to display data in the Interface Monitor. To display the messages in the Interface Monitor, you must maintain a recipient. Therefore, you reuse the already existing ALL_INTERFACES recipient. If your user is not assigned to the recipient, go to *System Configuration → Assign Recipients* and assign your user to the recipient.

6. Define the customer type as changeable.

Note:
If the IDoc contains an incorrect value for customer type, solve errors in Monitoring and Error Handling.

Use the following data:

Field	Value
Namespace	S###
Field path	E1SCU_CRE-E1BPSCUNEW-CUSTTYPE

a) Go to AIF Customizing (transaction `/AIF/CUST`) under *Error Handling → Namespace-Specific Features*.

b) Enter the namespace provided in the step.

c) Create a new entry for the IDoc interface in *Define namespace specific features*.

d) Go to AIF Customizing under *Error Handling → Interface-Specific Features* and enter your namespace. Choose *Define Changeable Fields* and create a new entry.

e) Enter an index and maintain the field path provided in the step.

f) Save.

7. To test the interface, create an IDoc.

Note:
Create the IDoc with the test transaction `WE19`.

Use the following data:

Field	Value
Basic Type	ZAIF_S###_FLCUST_CREATE01
EDIDC Details	
Port	**SAP<SYS-ID>** (replace SYS-ID with your system ID, for example, SAPT41)
Partner No. Sender	**ZAIF_S###**
Part. Type Sender	**LS**
Partner No. Receiver	**ZAIF_S###**
Part. TypeReceiver	**LS**
Message Type	**FLCUSTOMER_CREATEFROMDATA**
New Customer	
Custname	**Friedolin Hirsch**
Street	**Bachstelzenweg 1**
Postcode	**21614**
City	**Buxtehude**
Custtype	<empty>

a) Execute transaction `WE19`.

b) Choose *Basic Type* and enter the value in the corresponding field.

c) Execute.
An empty IDoc structure displays.

d) Double-click *EDIDC* and maintain the receiver, sender, and message type information.
Use the data provided in the EDIDC Details table.

e) Close the dialog box.

f) Expand *E1SCU_CRE* and double-click *E1BPSCUNEW*.
A dialog box appears.

g) To create a new customer, use the data provided in the New Customer table.

Note:
The value of the *CUSTTYPE* field creates an error in the action.

h) Close the dialog box.

i) Choose *Standard inbound* from the application toolbar.
An IDoc is created and sent. The IDoc is processed with AIF and your user is assigned to a corresponding recipient. The IDoc displays in the Interface Monitor.

8. Check the error messages.
   a) Start the Interface Monitor via transaction `/AIF/IFMON`.
   b) Choose and expand your namespace.
   Your *IDOC_SCEN2* interface displays as a sub node. Because the customer type is empty, an error occurs. The *Status* icon is red.
   c) Choose the icon for *Error Messages*.
   Monitoring and Error Handling starts. Monitoring and Error Handling is accessed from the Interface Monitor. Only the messages selected there display.
   d) Choose the message node.
   The application log messages written are displayed in *Application Log*. An error message indicates that the customer type is wrong.
   e) Choose structure *E1BPSCUNEW*.
   The structure of the interface displays in *data*.
   The transferrd data displays in *Data Content*. The structure contains the same data as entered in transaction `WE19`.
9. Solve the error.
   a) Scroll to the right and choose the *CUSTTYPE* field.
   b) Double-click the field.
   A dialog box appears.
   c) Enter **B** and close the dialog box.
   d) Save the data content.
   e) In *Data Messages*, choose *Restart*.
   f) Choose the *Read* button.
   After the error is solved and the message is reprocessed, the icon in *Data Messages* changes to green (tooltip: *Application document posted*). A message that the customer has been created will display in *Log Messages*.
   g) Note the customer number.
   h) Navigate to transaction `BC_GLOBAL_SCUST_DISP`.

i) Enter the customer number.

j) Choose *Display*.
The recently created customer displays.

Unit 8
Exercise 17

Monitor an IDoc Processed by AIF with a BAPI Call in Action - IDoc Scenario 3

Business Scenario

In this exercise, you learn how to monitor IDocs processed by AIF. The interface calls a BAPI in the AIF action. Because the structure of the BAPI and the IDoc differ, a structure mapping is required.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Preparation

This exercise requires some basic IDoc configurations. If you already have an existing IDoc, to change the processing from standard processing to AIF processing, adapt the following steps. The steps process the IDoc with AIF instead of using an IDoc standard functionality. They do not cover an IDoc's complete setup. They do not include creating a basic type and assigning a message type. These tasks are not part of AIF.

1. Assign a function module to a message type and basis type.

   If an IDoc is processed through AIF, assign one of the AIF function modules to the corresponding message type/IDoc basic type combination as an allowed function.

   /AIF/IDOC_INB_PROCESS_FUNC_0 handles one single IDoc over to AIF and can be used for Scenario 3.

   Note:
   The transaction to assign a function module to a message type is `WE57`.

   Use the following data:

Field	Value
Function module	/AIF/IDOC_INB_PROCESS_FUNC_0
Message type	ZAIF_T100_FLCUSTBOOK_CREATE
Basic IDoc type	ZAIF_S###_FLCUSTBOOK_CREATE01

2. Create an inbound process code.

   `/AIF/IDOC_INB_PROCESS_FUNC_0` can now be used for your IDoc. Create the needed processing code for the inbound processing with AIF.

Use the following data:

Field	Value
Process code	ZAIF_S###_FLCUSTBOOK_CREATE
Function Module	/AIF/IDOC_INB_PROCESS_FUNC_0

Note:
The transaction for IDoc inbound processing codes is `WE42`.

3. Maintain a partner profile.

   Use the following data:

Field	Value
Partner	LS
Logical system	Partner Type LS - ZAIF_S###
Message type	ZAIF_T100_FLCUSTBOOK_CREATE
Partner number	LS_AIF_S###
Inbound	
Process code	**ZAIF_S###_FLCUSTBOOK_CREATE**
Cancel Processing After Syntax Error	<Select>

Note:
Partner profiles are maintained in transaction `WE20`. They tell the system which processing code is used for an IDoc received from a special partner.

Task 2: Monitor an IDoc Processed by AIF

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc. Do this through the AIF IDoc generation report (transaction `/AIF/IDOC_GEN`).

1. Generate an IDoc structure.

   Use the following data:

Field	Value
General Details area	
Basic Type	**ZAIF_S###_FLCUSTBOOK_CREATE01**
Message Type	**ZAIF_T100_FLCUSTBOOK_CREATE**
Prefix Structure	**ZAIF_S###_**
Root Structure Name	**ZAIF_S###_FLCUSTBOOK_CREATE01**
Destination for Remote Search	<remains empty>

Field	Value
Create or update Structure	<flag>
Create Interface	<flag>, when flagged hit *Enter* on your keyboard
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**
Interface Name	**IDOC_SCEN3**
Interface Version	**1**
IDoc Processing Scenario	**03** (AIF Runtime; User specific Action)
Customizing Request	<the one given to you from the instructor>

2. Check in the AIF Customizing the structures created from the IDoc Generator.

Note:
The structures can also be found in your package ZAIF_SXXX.

Use the following data:

Component	Type
EDIDC	**STRUC**
E1BPSCUNEW	**STRUC**
.INCLUDE(E1BPSCUNEW)	
E1BPSBONEW	**STRUC**

3. Check the created interface.

Check the interface created from the IDoc generation report. Check the engines chosen from the report. If you find a different result, change the settings to the settings in the solution of this exercise.

This time, you do not use the IDoc standard function in your action. The IDoc generation report cannot suggest the SAP structure. Add it manually. In this example, the BAPI you use needs the structure `/AIF/TEST_SAP_FLBOOKING` as the SAP structure.

Use the following data:

Field	Value
Namespace	**S###**

Field	Value
Interface Name	**IDOC_SCEN3**
Interface Version	**1**
SAP Data Structure	<is empty>
Raw Data Structure	**ZAIF_S###_FLCUSTBOOK_CREATE01**
after check	
SAP data structure	/AIF/TEST_SAP_FLBOOKING
Engine Details	
Application Engine ID	**IDoc**
Persistency Engine ID	**IDoc**
Selection Engine ID	**AIF Index Table**
Logging Engine ID	**AIF Application Log**

Task 3: Create Further Objects

1. Create a structure mapping.

> Note:
> For scenario 3, the structures of the IDoc and the structure of the BAPI are different. This time, you need a mapping.

Use the following data:

Field in Destination Structure	Field Name 1
Source Structure	<space>
Number of structure mapping	10
CUSTOMER_DATA-CUSTNAME	*E1BPSCUNEW-CUSTNAME*
CUSTOMER_DATA-FORM	*E1BPSCUNEW-FORM*
CUSTOMER_DATA-STREET	*E1BPSCUNEW-STREET*
CUSTOMER_DATA-CUSTTYPE	*E1BPSCUNEW-CUSTTYPE*
Second Mappings	
BOOKING_DATA-CLASS	**E1BPSCUNEW-E1BPSBONEW-CLASS**
BOOKING_DATA-AIRLINEID	**E1BPSCUNEW-E1BPSBONEW-AIRLINEID**
BOOKING_DATA-CONNECTID	**E1BPSCUNEW-E1BPSBONEW-CONNECTID**
BOOKING_DATA-AGENCYNUM	**E1BPSCUNEW-E1BPSBONEW-AGENCYNUM**
BOOKING_DATA-FLIGHTDATE	**E1BPSCUNEW-E1BPSBONEW-FLIGHTDATE**

2. Create a check for customer type.

The customer type and customer name are mandatory fields. You want to prove that they contain content in the RAW structure by creating a check for every field. As the error message class use ZAIF_TRAINING. Use the appropriate message number.

Field	Value
First Check Assignment	
Number of the Check	**10**
Namespace	**S###**
Check	**CK_CUSTTYPE** - Press *Enter*. A dialog box will appear asking if you want to create the check. Confirm.
Check Raw data	**X**
Ignore Data if Check is not successful	**Treat as error if check is not successful**
Field Name 1	**E1BPSCUNEW-CUSTTYPE**
Message	
Check Description	**Check Customer Type (S###)**
Error Msg. Class	**ZAIF_TRAINING**
Error Msg. Number	**006**
Variable Definition	**$1** (at runtime $1 will be replaced with the value of Field Name 1 from the check assignment)
Allowed characters	BP

3. Create a check for customer name.

   The customer type and customer name are mandatory fields. You want to prove that they contain content in the RAW structure by creating a check for every field. As the error message class use ZAIF_TRAINING. Use the appropriate message number.

Second Check Assignment	Value
Number of the Check	**20**
Namespace	**S###**
Check	**CK_CUSTNAME** - Press *Enter*. A dialog box will appear asking if you want to create the check. Confirm.
Check Raw Data	**X**
Ignore Data if Check is not successful	**Treat as error if check is not successful**
Field Name 1	**E1BPSCUNEW-CUSTNAME**
Check Customer Name	

Second Check Assignment	Value
Check Description	**Check Customer Name (S###)**
Error Msg. Class	**ZAIF_TRAINING**
Error Msg. Number	**007**

4. Create an action.

   Use the following data:

Field	Value
Action	AC_FLCUST_BOOK_CREA
Action description	**Create customer and booking (S###)**
Commit Mode	**COMMIT WORK**
Commit Level	**After Each Function**
Main Component Type	**/AIF/TEST_SAP_FLBOOKING**

   You want to post the data that is delivered from the IDoc with an BAPI instead of the standard IDoc function. Create an action called *AC_FLCUST_BOOK_CREA*. It uses the main component type */AIF/TEST_SAP_FLBOOKING* that you already used as the SAP structure in the interface definition. You want to commit your work after every function. You need one function for the posting of the customer.

5. Define a function that can create a new customer for your action.

   Use the following data:

Field	Value
Function number	10
Function module	/AIF/TEST_AC_FLCUST_CREATE

6. As in your own interface FLBOOKING/1, two different function modules create the customer and the flight booking. So again you need the *CUSTOMERNUMBER* as a restorable field in case something goes wrong.

7. To post the flight booking, you need a second function. You put it as a second function inside of the same action.

   Use the following data:

Field	Value
Function number	20
Function module	/AIF/TEST_AC_FLBOOKING_CREATE

8. Assign the action you created to the structure mapping of your interface IDOC_SCEN3/1.

   Use the following data:

Field	Value
Action Number	**10**
Namespace	**S###**
Action	**AC_FLCUST_BOOK_CREA**

9. Maintain a recipient.

   During the processing of the IDoc, AIF is called and index table and message index table entries are written. You can display data in the Interface Monitor. To display the messages in the Interface Monitor, a recipient has to be maintained.

   If your user is still assigned to the ALL_INTERFACES recipient, you can skip this step. If your user is not assigned to the recipient, you have to add it again.

10. Define customer name and type as changeable.

    You want to be able to solve errors in Monitoring and Error Handling if the IDoc contains an incorrect value for customer name or type. Create the corresponding editable fields.

Task 4: Test Your Development

1. Test your development.

   Use the following data:

Field	Value
IDoc Header	
Basic Type	ZAIF_S###_FLCUSTBOOK_CREATE01
Message Type	**ZAIF_T100_FLCUSTBOOK_CREATE**
Port	**SAP<SYS-ID>** (replace SYS-ID with your system ID, for example, **SAPT41**)
Partner No. Sender	**ZAIF_S###**
Part. Type Sender	**LS**
Partner No. Receiver	**ZAIF_S###**
Part. Type Receiver	**LS**
Flight Booking Data	
Airline ID	**LH**
Connection ID	**0400**
Flight Date	**20140412**
Class	**Y**
Agency	**109**
Valid customer type	B

Note:
In order to test the interface you require an IDoc. Use the IDoc test transaction to create one.

Unit 8
Solution 17

Monitor an IDoc Processed by AIF with a BAPI Call in Action - IDoc Scenario 3

Business Scenario

In this exercise, you learn how to monitor IDocs processed by AIF. The interface calls a BAPI in the AIF action. Because the structure of the BAPI and the IDoc differ, a structure mapping is required.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Preparation

This exercise requires some basic IDoc configurations. If you already have an existing IDoc, to change the processing from standard processing to AIF processing, adapt the following steps. The steps process the IDoc with AIF instead of using an IDoc standard functionality. They do not cover an IDoc's complete setup. They do not include creating a basic type and assigning a message type. These tasks are not part of AIF.

1. Assign a function module to a message type and basis type.

   If an IDoc is processed through AIF, assign one of the AIF function modules to the corresponding message type/IDoc basic type combination as an allowed function.

   /AIF/IDOC_INB_PROCESS_FUNC_0 handles one single IDoc over to AIF and can be used for Scenario 3.

   Note:
   The transaction to assign a function module to a message type is `WE57`.

   Use the following data:

Field	Value
Function module	/AIF/IDOC_INB_PROCESS_FUNC_0
Message type	ZAIF_T100_FLCUSTBOOK_CREATE
Basic IDoc type	ZAIF_S###_FLCUSTBOOK_CREATE01

   a) Navigate to transaction `WE57` and switch to change mode.

   b) Choose *New Entries*.

c) Enter the function module provided in the step.

d) Choose *Function Module* as the function type.

e) Enter the value of the message type and the basic IDoc type provided in the step.

f) Choose *Direction Inbound*.

g) Save.

Note:
If prompted, use your transport request.

2. Create an inbound process code.

`/AIF/IDOC_INB_PROCESS_FUNC_0` can now be used for your IDoc. Create the needed processing code for the inbound processing with AIF.

Use the following data:

Field	Value
Process code	ZAIF_S###_FLCUSTBOOK_CREATE
Function Module	/AIF/IDOC_INB_PROCESS_FUNC_0

Note:
The transaction for IDoc inbound processing codes is `WE42`.

a) Navigate to transaction `WE42` and switch to change mode.

b) Create a new entry.

c) Enter the value provided in the step for *Process Code*.

d) Enter a description.

e) Choose *Processing with ALE service* and *Processing by function module*.

f) Choose *Enter* and save.
A new window opens.

g) In the *Function Module* field, choose the value provided in the step for the function module.

h) Save.

3. Maintain a partner profile.

Use the following data:

Field	Value
Partner	LS
Logical system	Partner Type LS - ZAIF_S###
Message type	ZAIF_T100_FLCUSTBOOK_CREATE

Field	Value
Partner number	LS_AIF_S###
Inbound	
Process code	**ZAIF_S###_FLCUSTBOOK_CREATE**
Cancel Processing After Syntax Error	<Select>

Note:
Partner profiles are maintained in transaction `WE20`. They tell the system which processing code is used for an IDoc received from a special partner.

a) Navigate to `WE20` and open the node *Logical Systems*.

b) Choose the logical system provided in the step.

c) Choose *Create inbound parameters*.

d) Enter the message type provided in the step.

e) In *Inbound Options*, enter the data from the Inbound table.

f) Choose *Trigger Immediately*.

g) Save.

Task 2: Monitor an IDoc Processed by AIF

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc. Do this through the AIF IDoc generation report (transaction `/AIF/IDOC_GEN`).

1. Generate an IDoc structure.

   Use the following data:

Field	Value
General Details area	
Basic Type	**ZAIF_S###_FLCUSTBOOK_CREATE01**
Message Type	**ZAIF_T100_FLCUSTBOOK_CREATE**
Prefix Structure	**ZAIF_S###_**
Root Structure Name	**ZAIF_S###_FLCUSTBOOK_CREATE01**
Destination for Remote Search	<remains empty>
Create or update Structure	<flag>
Create Interface	<flag>, when flagged hit *Enter* on your keyboard
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###

Field	Value
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**
Interface Name	**IDOC_SCEN3**
Interface Version	**1**
IDoc Processing Scenario	**03** (AIF Runtime; User specific Action)
Customizing Request	<the one given to you from the instructor>

a) Navigate to transaction `/AIF/IDOC_GEN`.

b) Enter the data from the IDoc Structure table.

c) Enter your transport request in the *Workbench Request/Task* field.

d) Choose *Execute*.
A structure for your basic type is generated. An interface in your namespace is also created.

2. Check in the AIF Customizing the structures created from the IDoc Generator.

Note:
The structures can also be found in your package ZAIF_SXXX.

Use the following data:

Component	Type
EDIDC	**STRUC**
E1BPSCUNEW	**STRUC**
.INCLUDE(E1BPSCUNEW)	
E1BPSBONEW	**STRUC**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Interfaces*.

b) Choose your namespace and open your interface (*IDOC_SCEN3*).

c) Double-click the raw data structure.
A new window opens and the structure displays (transaction `SE11` is started).

d) Using the data in the Structure table, check the structure and substructure.

Note:
Instead of using the forward navigation, you can also access transaction `SE11` directly to display your generated structure.

e) Execute transaction `SE80` and check if the generated structures are assigned to your package.

Note:
If you cannot see the structure and table types in your package, right-click the package. Choose *Other Functions* → *Rebuild Object List*.

3. Check the created interface.

Check the interface created from the IDoc generation report. Check the engines chosen from the report. If you find a different result, change the settings to the settings in the solution of this exercise.

This time, you do not use the IDoc standard function in your action. The IDoc generation report cannot suggest the SAP structure. Add it manually. In this example, the BAPI you use needs the structure `/AIF/TEST_SAP_FLBOOKING` as the SAP structure.

Use the following data:

Field	Value
Namespace	**S###**
Interface Name	**IDOC_SCEN3**
Interface Version	**1**
SAP Data Structure	<is empty>
Raw Data Structure	**ZAIF_S###_FLCUSTBOOK_CREATE01**
after check	
SAP data structure	/AIF/TEST_SAP_FLBOOKING
Engine Details	
Application Engine ID	**IDoc**
Persistency Engine ID	**IDoc**
Selection Engine ID	**AIF Index Table**
Logging Engine ID	**AIF Application Log**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development* → *Define Interfaces*.

b) Enter your namespace.

c) Check that the interface, created by the report, exists.

d) Check that the interface details correspond to the data from the table above.
A BAPI is used in this scenario to create a customer and a flight booking.

The SAP structure field is empty.

e) Enhance the interface definition and add the value of the SAP data structure provided in the table above, below the row **after check**.

f) Save.

g) Return to AIF Customizing and navigate to *Interface Development → Additional Interface Properties → Specific Interface Engines*.

h) Enter your namespace and choose your interface *IDOC_SCEN3*.

i) Check the interface details correspond to the data in the table above, below the row **Engine Details**.

Task 3: Create Further Objects

1. Create a structure mapping.

Note:
For scenario 3, the structures of the IDoc and the structure of the BAPI are different. This time, you need a mapping.

Use the following data:

Field in Destination Structure	Field Name 1
Source Structure	<space>
Number of structure mapping	10
CUSTOMER_DATA-CUSTNAME	*E1BPSCUNEW-CUSTNAME*
CUSTOMER_DATA-FORM	*E1BPSCUNEW-FORM*
CUSTOMER_DATA-STREET	*E1BPSCUNEW-STREET*
CUSTOMER_DATA-CUSTTYPE	*E1BPSCUNEW-CUSTTYPE*
Second Mappings	
BOOKING_DATA-CLASS	**E1BPSCUNEW-E1BPSBONEW-CLASS**
BOOKING_DATA-AIRLINEID	**E1BPSCUNEW-E1BPSBONEW-AIRLINEID**
BOOKING_DATA-CONNECTID	**E1BPSCUNEW-E1BPSBONEW-CONNECTID**
BOOKING_DATA-AGENCYNUM	**E1BPSCUNEW-E1BPSBONEW-AGENCYNUM**
BOOKING_DATA-FLIGHTDATE	**E1BPSCUNEW-E1BPSBONEW-FLIGHTDATE**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Structure Mappings*.

b) Choose your namespace, interface, and version.

c) To create a mapping on root level, enter the value provided in the step for *Source Structure* and choose *Enter*.

d) Choose the newly created (empty) line and, in *Assign Destination Structure*, create a new entry.

e) Enter the value provided in the step for the number of structure mapping.

f) In the *Destination Structure* field, use the F4 help to choose the root structure. The field remains empty.

g) Choose *Define Field Mappings* on the left-hand side and create a new entry.

h) In *Destination Structure*, choose the F4 help, choose *CUSTOMER_DATA* and choose *CUSTNAME*.

i) In *Field Name 1*, choose the F4 help, choose *E1BPSCUNEW*, and choose *CUSTNAME*.

j) To create a further mapping, choose *Next entry* and, for the nodes CUSTOMER_DATA and E1BPSCUNEW, enter the data from the First Mappings table.

k) In *Field in Destination Structure*, choose *BOOKING_DATA* and choose the *AIRLINEID* field. In *Field Name 1*, choose the F4 help, choose *E1BPSBONEW*, and choose *AIRLINEID*.

l) For the nodes BOOKING_DATA and E1BPSCUNEW-E1BPSBONEW, enter the data from the Second Mappings table and save.

2. Create a check for customer type.

The customer type and customer name are mandatory fields. You want to prove that they contain content in the RAW structure by creating a check for every field. As the error message class use ZAIF_TRAINING. Use the appropriate message number.

Field	Value
First Check Assignment	
Number of the Check	**10**
Namespace	**S###**
Check	**CK_CUSTTYPE** - Press *Enter*. A dialog box will appear asking if you want to create the check. Confirm.
Check Raw data	**X**
Ignore Data if Check is not successful	**Treat as error if check is not successful**
Field Name 1	**E1BPSCUNEW-CUSTTYPE**
Message	
Check Description	**Check Customer Type (S###)**
Error Msg. Class	**ZAIF_TRAINING**
Error Msg. Number	**006**

Field	Value
Variable Definition	**$1** (at runtime $1 will be replaced with the value of Field Name 1 from the check assignment)
Allowed characters	BP

a) To create a new entry, navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development* → *Define Structure Mapping*. Choose your namespace, interface, and version. Choose the empty line in *Source Structure* and navigate to *Assign Checks*.

b) Enter the data from the First Check Assignment table and save.

c) Double-click the check and, to fill in the message number in case of error, switch to change mode.

d) Enter the data from the table above, below message.

e) In the left hand side menu, choose *Define Single Check*. Create a new entry. Enter **10** as number of check. In the *Field Check* field, choose **Not Empty**. Enter the value provided in the step for allowed characters and save.

3. Create a check for customer name.

   The customer type and customer name are mandatory fields. You want to prove that they contain content in the RAW structure by creating a check for every field. As the error message class use ZAIF_TRAINING. Use the appropriate message number.

Second Check Assignment	Value
Number of the Check	**20**
Namespace	**S###**
Check	**CK_CUSTNAME** - Press *Enter*. A dialog box will appear asking if you want to create the check. Confirm.
Check Raw Data	**X**
Ignore Data if Check is not successful	**Treat as error if check is not successful**
Field Name 1	**E1BPSCUNEW-CUSTNAME**
Check Customer Name	
Check Description	**Check Customer Name (S###)**
Error Msg. Class	**ZAIF_TRAINING**
Error Msg. Number	**007**

a) Return to the Customizing activity *Assign Checks* and choose the next entry.

b) For the next check, use the data from the Second Check Assignment table and save.

c) Double-click *Check* and switch to change mode.

d) Enter the data from the table above, below **Check Customer Name**.

e) In the left-hand side menu, choose *Define Single Check*. Create a new entry. Enter **`10`** as the number of check. In the *Field Check* field, choose **`Not Empty`**.

f) Save and close the window.

4. Create an action.

Use the following data:

Field	Value
Action	AC_FLCUST_BOOK_CREA
Action description	**`Create customer and booking (S###)`**
Commit Mode	**`COMMIT WORK`**
Commit Level	**`After Each Function`**
Main Component Type	**`/AIF/TEST_SAP_FLBOOKING`**

You want to post the data that is delivered from the IDoc with an BAPI instead of the standard IDoc function. Create an action called *AC_FLCUST_BOOK_CREA*. It uses the main component type */AIF/TEST_SAP_FLBOOKING* that you already used as the SAP structure in the interface definition. You want to commit your work after every function. You need one function for the posting of the customer.

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Actions*.

b) Enter your namespace.

c) Create a new entry.

d) Enter the value of *Action* provided in the step and choose *Enter*.

e) Maintain values from the data provided in the Action table.

5. Define a function that can create a new customer for your action.

Use the following data:

Field	Value
Function number	10
Function module	/AIF/TEST_AC_FLCUST_CREATE

a) Navigate to *Define Functions* in the tree menu on the left side and create a new entry.

b) Enter the data provided in the table above.

c) Save.

6. As in your own interface FLBOOKING/1, two different function modules create the customer and the flight booking. So again you need the *CUSTOMERNUMBER* as a restorable field in case something goes wrong.

a) Create the field to restore for the *CUSTOMERNUMBER* field.

b) In *Define Functions*, choose your newly created function and double-click *Define Fields to Restore* on the left-hand side.

c) Create a new entry. Enter **`CUSTOMERNUMBER`** as the field name or use the F4 help.

d) Save.

7. To post the flight booking, you need a second function. You put it as a second function inside of the same action.

   Use the following data:

Field	Value
Function number	20
Function module	/AIF/TEST_AC_FLBOOKING_CREATE

a) Return to *Define Functions* to create a second function.

Note:
This function can create a flight booking for the newly created customer.

b) Enter the data from the table above.

c) Save.

8. Assign the action you created to the structure mapping of your interface IDOC_SCEN3/1.

   Use the following data:

Field	Value
Action Number	**`10`**
Namespace	**`S###`**
Action	**`AC_FLCUST_BOOK_CREA`**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development* → *Define Structure Mappings*. Choose your namespace, interface, and version.

b) Assign the action to your structure mapping.

c) Maintain the data provided in the table above.

9. Maintain a recipient.

   During the processing of the IDoc, AIF is called and index table and message index table entries are written. You can display data in the Interface Monitor. To display the messages in the Interface Monitor, a recipient has to be maintained.

   If your user is still assigned to the ALL_INTERFACES recipient, you can skip this step. If your user is not assigned to the recipient, you have to add it again.

a) Navigate to *System Configuration* → *Assign Recipients* and assign your user to the recipient.

10. Define customer name and type as changeable.

    You want to be able to solve errors in Monitoring and Error Handling if the IDoc contains an incorrect value for customer name or type. Create the corresponding editable fields.

    a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Error Handling → Namespace-Specific Features*.

    b) Enter your namespace **S###**.

    c) Create a new entry for the IDoc interface in *Define namespace specific features*.

    d) Navigate to AIF Customizing under *Error Handling → Interface-Specific Features* and enter your namespace, interface, and version.

    e) Choose *Define Changeable Fields* and create a new entry. Enter **10** as the index and enter the field path **E1BPSCUNEW-CUSTNAME**.

    f) Create a second entry, enter **20** as the index, and maintain the field path **E1BPSCUNEW-CUSTTYPE**.

    g) Save.

Task 4: Test Your Development

1. Test your development.

   Use the following data:

Field	Value
IDoc Header	
Basic Type	ZAIF_S###_FLCUSTBOOK_CREATE01
Message Type	**ZAIF_T100_FLCUSTBOOK_CREATE**
Port	**SAP<SYS-ID>** (replace SYS-ID with your system ID, for example, **SAPT41**)
Partner No. Sender	**ZAIF_S###**
Part. Type Sender	**LS**
Partner No. Receiver	**ZAIF_S###**
Part. Type Receiver	**LS**
Flight Booking Data	
Airline ID	**LH**
Connection ID	**0400**
Flight Date	**20140412**
Class	**Y**
Agency	**109**
Valid customer type	B

Note:
In order to test the interface you require an IDoc. Use the IDoc test transaction to create one.

a) In transaction `WE19`, enter the *Basic Type* value provided in the step. Execute.
An empty IDoc structure displays.

b) Double-click *EDIDC*. Maintain the data from the IDoc Header table.

c) Close the dialog box and double-click *E1BPSCUNEW*. A dialog box appears. Maintain data to create a new customer. Maintain at least the following fields: *Form*, *Street*, *Postcode*, and *City*.

Note:
In order to create an error in the action, ensure that fields *CUSTTYPE* and *CUSTNAME* are empty.

d) Close the dialog box and expand the node. Maintain the flight booking data with the data provided in the Flight Booking Data table.

Note:
It is possible that the flight data described above is not working, for example, if the flight is completely occupied. In this case, you have to go to the table SFLIGHT and choose flights where seats are still available and change your test data accordingly.

e) Close the dialog and choose *Standard inbound* from the application toolbar. An IDoc is created and sent. Note down the IDoc number that displays after the IDoc was transferred to the application.
The IDoc was processed with AIF and your user is assigned to a corresponding recipient. The IDoc will be displayed in the Interface Monitor. Check it and find out if an error occurred. If you find an error edit the message and restart it. Check if it worked this time.

f) Start the Interface Monitor via transaction `/AIF/IFMON`. On the right side, choose and expand your namespace.
Your *IDOC_SCEN3* interface displays as a sub-node. Because the customer type and name were empty, an error should have occurred. The *Status* icon should be red.

g) Choose the *Error Messages* icon.

Monitoring and Error Handling is accessed from the Interface Monitor. Only the messages selected there are displayed.

Monitoring and Error handling starts.

h) Choose the message node. The structure of the interface displays in the *Data Structure* view.
The application log messages written are displayed in the application log view. Two error messages are displayed: **`Customer type is not valid. Customer type`**

`has to be B or P` and **`Customer name is empty. Customer cannot be created`**.

i) Double-click the error message related to the customer type. You are forwarded to the field where the error occurred in *Data Content*. Double-click the field containing the error. A dialog box appears. Insert the valid customer type value provided in the step.

j) Double-click the error message related to the customer name. You are forwarded to the field where the error occurred in *Data Content*. Double-click on the erroneous field and enter a customer name in the dialog box. Save.

k) In *Data Messages*, choose *Restart*, and choose *Read*.
After the error is solved and the message is reprocessed, the icon in the *Data Messages* view changes to green (tooltip: *Application document posted*). In the *Log Messages* view, a message that the customer has been created and the flight was booked displays. Note the customer and the flight number.

l) Check the customer. Navigate to transaction `BC_GLOBAL_SCUST_DISP`. Enter your customer number and choose *Display*.
The newly created customer displays.

m) Check the flight booking. Navigate to transaction `BC_GLOBAL_SBOOK_DISP` and enter the airline ID and the booking number of your flight booking.
Your flight booking displays.

Unit 8
Exercise 18

Monitor an IDoc with AIF Enabler - IDoc Scenario 4

Business Scenario

In this exercise, you learn how to monitor IDocs only processed by ALE Runtime. To use some advanced features of AIF, such as index tables and alerting, the AIF Enabler is called during processing. This exercise demonstrates the enabler scenario without using the application log.

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc. For IDoc scenarios, the message type and IDoc basic type must be set for the interface. Do this from the IDoc generation report. Check the results in the additional interface settings of AIF Customizing of your interface.

The AIF Enabler will write index table entries. You want to create an interface-specific index table to store those entries. This index table enables a search for a key field in Monitoring and Error Handling. You want a field *SCUSTNO* as an additional field and the data element used is *S_CUSTOMER*. You can use the AIF Standard single index table as a template.

Selecting your key fields from your new index table requires an interface-specific selection screen. Create a selection screen with the customer number as input parameter. Name the report.

During the processing of the IDoc, AIF is called and index table and message index table entries are written. You can display data in the Interface Monitor. To display the messages in the Interface Monitor, you must maintain a recipient. You want to solve errors in Monitoring and Error Handling if the IDoc contains an incorrect value for customer number. To do this, create the corresponding editable field.

At the end, you want to test your development. You create a new IDoc via transaction `WE19` and change the customer you created in the previous exercise, IDoc Scenario 2 – Process an IDoc by AIF With IDoc Function Call in Action.

Send a second IDoc using a non-existing customer number.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Monitor an IDoc with AIF Enabler

1. Create a DDIC structure out of the basic type of the IDoc.

   Use the following data:

Field	Value
General Details area	

Field	Value
Basic Type	**ZAIF_S###_FLCUST_CHANGE01**
Message Type	**FLIGHTCUSTOMER_CHANGE**
Structure Prefix	**ZAIF_S###_**
Root Structure Name	**ZAIF_S###_FLCUST_CHANGE01**
Destination for Remote Search	<remains empty>
Create or update Structure	<flag>
Create Interface	<flag>, when flagged, hit *Enter* on your keyboard
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**
Interface Name	**IDOC_SCEN4**
Interface Version	**1**
IDoc Processing Scenario	**05** (AIF Enabler; no Application Log)
Interface Description	**Interface for IDoc Scenario 4**

2. Check in the AIF Customizing the structures created from the IDoc Generator.

Note:
The structures can also be found in your package ZAIF_S###.

Compare to the following data:

Component	Type
EDIDC	**STRUC**
E1SCU_CHG	**STRUC**
.INCLUDE(E1SCU_CHG)	
E1BPSCUNEW	**STRUC**
E1BPSCUNEX	
E1BPPAREX	**TAB**

3. Check the interface created from the IDoc generation report. Check the engines chosen from the report.

Note:
If you find a different result, change the settings to the settings in the solution of this exercise.

Use the following data:

Field	Value
Namespace	**S###**
Interface Name	**IDOC_SCEN4**
Interface Version	**1**
SAP Data Structure	**ZAIF_S###_FLCUST_CHANGE01**
Raw Data Structure	**ZAIF_S###_FLCUST_CHANGE01**
Move Corresponding Structures	X
Engines	
Application Engine ID	IDoc
Persistency Engine ID	IDoc
Selection Engine ID	AIF Index Table
Logging Engine ID	IDoc Status Records

4. Set the message type and IDoc basic type for the interface. Check the results in the additional interface settings of AIF Customizing of your interface.

   Use the following data:

Field	Value
Message Type	*FLIGHTCUSTOMER_CHANGE*
Basic Type	*ZAIF_S###_FLCUST_CHANGE01*

5. Create an interface-specific index table to store index table entries.

   Use the following data:

   Copy to table: **ZS###_IDX_4IDOC**

Field	Data Element
SCUSTNO	S_CUSTOMER

6. Create a selection screen that has the customer number as input parameter.

Note:
The screen must be a subscreen. The program type is module pool for this and the screen must be programmed. You need variables and select-options or parameters for every field that is not part of the standard single index table.

Use the following data:

Field	Value
Report name	ZAIF_S###_4IDOC_SEL_SCREEN
Title	**Selection Screen for IDoc Interface with Enabler (S###)**
Dictionary Ref.	<select>

7. To change the single index table from the standard, assign it to your interface in the namespace-specific features, and add your program and screen number for the selection.

   Use the following data:

Field	Value
Interface	**IDOC_SCEN4**
Version	**1**
Message Index Table	**ZS###_IDX_4IDOC**
Program Name	**ZAIF_S###_4IDOC_SEL_SCREEN**
Screen Number	**0001**

8. To fill the interface-specific fields of your index table, the system needs the interface field providing them.

Note:
Use the interface-specific features to add the necessary information.

Use the following data:

Field	Value
Key Field	**SCUSTNO**
Data Element	**S_CUSTOMER**
Select-Options / Parameter	**P_CUSTNO** (use the parameter name created in the selection screen)
Field name	**E1SCU_CHG-CUSTOMERNUMBER**
Icon	**@A0@**
Tooltip	**Customer**

9. Optional: Maintain a recipient.

   If your user is still assigned to the ALL_INTERFACES recipient, you can skip this step. If your user is not assigned to the recipient, you must add it again.

10. To solve errors in Monitoring and Error Handling, create a corresponding editable field.

    Use the following data:

    Index: **10**

Field Path: **E1SCU_CHG-CUSTOMERNUMBER**

Task 2: Test Your Development

1. Test your development by creating a new IDoc via transaction `WE19` and changing the customer.

Note:
Ensure you fill the corresponding X-Structure. Send a second IDoc using a non-existing customer number.

Use the following data:

Field	Value
Segment	E1SCU_CHG
Basic Type	ZAIF_S###_FLCUST_CHANGE01
EDIDC fields	
Port	**SAP<SYS-ID>** (replace SYS-ID with your system ID) for example, SAPT41
Partner No. Sender	**ZAIF_S###**
Part. Type Sender	**LS**
Partner No. Receiver	**ZAIF_S###**
Part. Type Receiver	**LS**
Message Type	**FLIGHTCUSTOMER_CHANGE**
E1BPSCUNEW fields	
Custname	**Fridolin Hirsch**
Street	**Bachstelzenweg 1**
Postcode	**47111**
City	**Changed city**

Unit 8
Solution 18

Monitor an IDoc with AIF Enabler - IDoc Scenario 4

Business Scenario

In this exercise, you learn how to monitor IDocs only processed by ALE Runtime. To use some advanced features of AIF, such as index tables and alerting, the AIF Enabler is called during processing. This exercise demonstrates the enabler scenario without using the application log.

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc. For IDoc scenarios, the message type and IDoc basic type must be set for the interface. Do this from the IDoc generation report. Check the results in the additional interface settings of AIF Customizing of your interface.

The AIF Enabler will write index table entries. You want to create an interface-specific index table to store those entries. This index table enables a search for a key field in Monitoring and Error Handling. You want a field *SCUSTNO* as an additional field and the data element used is *S_CUSTOMER*. You can use the AIF Standard single index table as a template.

Selecting your key fields from your new index table requires an interface-specific selection screen. Create a selection screen with the customer number as input parameter. Name the report.

During the processing of the IDoc, AIF is called and index table and message index table entries are written. You can display data in the Interface Monitor. To display the messages in the Interface Monitor, you must maintain a recipient. You want to solve errors in Monitoring and Error Handling if the IDoc contains an incorrect value for customer number. To do this, create the corresponding editable field.

At the end, you want to test your development. You create a new IDoc via transaction `WE19` and change the customer you created in the previous exercise, IDoc Scenario 2 – Process an IDoc by AIF With IDoc Function Call in Action.

Send a second IDoc using a non-existing customer number.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Monitor an IDoc with AIF Enabler

1. Create a DDIC structure out of the basic type of the IDoc.

   Use the following data:

Field	Value
General Details area	

Field	Value
Basic Type	**ZAIF_S###_FLCUST_CHANGE01**
Message Type	**FLIGHTCUSTOMER_CHANGE**
Structure Prefix	**ZAIF_S###_**
Root Structure Name	**ZAIF_S###_FLCUST_CHANGE01**
Destination for Remote Search	<remains empty>
Create or update Structure	<flag>
Create Interface	<flag>, when flagged, hit *Enter* on your keyboard
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**
Interface Name	**IDOC_SCEN4**
Interface Version	**1**
IDoc Processing Scenario	**05** (AIF Enabler; no Application Log)
Interface Description	**Interface for IDoc Scenario 4**

a) Navigate to the Generate IDoc Structure and Interface Definition report (transaction `/AIF/IDOC_GEN`).

b) Enter the data from the Generate IDoc Structure and Interface Definition.

c) Enter your transport request in the Workbench *Request/Task* field.

d) Execute.
A structure for your basic type is generated. An interface in your namespace is also created.

2. Check in the AIF Customizing the structures created from the IDoc Generator.

Note:
The structures can also be found in your package ZAIF_S###.

Compare to the following data:

Component	Type
EDIDC	**STRUC**
E1SCU_CHG	**STRUC**

Component	Type
.INCLUDE(E1SCU_CHG)	
E1BPSCUNEW	**STRUC**
E1BPSCUNEX	
E1BPPAREX	**TAB**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Interfaces*.

b) Choose your namespace and open your interface (*IDOC_SCEN4*).

c) Double-click on the raw data structure.
A new window opens and the structure displays (transaction `SE11` is started). The structure and substructure has the data provided in the Raw Data Structure table.

d) Instead of using the forward navigation, you can also access transaction **SE11** directly to display your generated structure.

e) Call transaction `SE80` and check if the generated structures are assigned to your package. If you cannot see the structure and table types in your package, right-click the package and choose *Other Functions → Rebuild Object List*.

3. Check the interface created from the IDoc generation report. Check the engines chosen from the report.

Note:
If you find a different result, change the settings to the settings in the solution of this exercise.

Use the following data:

Field	Value
Namespace	**S###**
Interface Name	**IDOC_SCEN4**
Interface Version	**1**
SAP Data Structure	**ZAIF_S###_FLCUST_CHANGE01**
Raw Data Structure	**ZAIF_S###_FLCUST_CHANGE01**
Move Corresponding Structures	X
Engines	
Application Engine ID	IDoc
Persistency Engine ID	IDoc
Selection Engine ID	AIF Index Table
Logging Engine ID	IDoc Status Records

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Interfaces*. Enter your namespace.

b) Check, that the interface, created by the report, exists and contains the data shown in the table above.

c) Return to AIF Customizing (transaction `/AIF/CUST`) and choose *Interface Development → Additional Interface Properties → Specific Interface Engines*. Enter your namespace and select your *IDOC_SCEN4* interface. Check the data shown in the Engines table are maintained.

d) Save.

4. Set the message type and IDoc basic type for the interface. Check the results in the additional interface settings of AIF Customizing of your interface.

   Use the following data:

Field	Value
Message Type	*FLIGHTCUSTOMER_CHANGE*
Basic Type	*ZAIF_S###_FLCUST_CHANGE01*

   a) Call transaction `/AIF/CUST` and choose *Interface Development → Additional Interface Properties → Assign IDoc Types*.

   b) Choose your namespace and check the entry for your interface.

   c) Check the data from the Message Type and Basic Type table are assigned.

5. Create an interface-specific index table to store index table entries.

   Use the following data:

   Copy to table: **`ZS###_IDX_4IDOC`**

Field	Data Element
SCUSTNO	S_CUSTOMER

   a) In transaction `SE11`, enter database table **`/AIF/STD_IDX_TBL`** and choose *Copy*.

   b) Change the name of the *Copy to table* field to the value provided in the table.

   c) Choose *OK*.

   d) Enter your package, choose your transport request, and save.

   e) Open the table and switch to edit mode.

   f) Add a new field using the data from the New Field table.

   g) Save and activate the table.

6. Create a selection screen that has the customer number as input parameter.

Note:
The screen must be a subscreen. The program type is module pool for this and the screen must be programmed. You need variables and select-options or parameters for every field that is not part of the standard single index table.

Use the following data:

Field	Value
Report name	ZAIF_S###_4IDOC_SEL_SCREEN
Title	**Selection Screen for IDoc Interface with Enabler (S###)**
Dictionary Ref.	<select>

a) In transaction `SE38`, enter the report name provided in the table above and choose *Create*.

b) Enter the value provided in the table above for *Title*.

c) Choose the type *Module Pool*. Save.

d) Enter your package and save the program to your transport request.

The code resembles the following snippet:

```
SELECTION-SCREEN BEGIN OF SCREEN 0001 AS SUBSCREEN.
PARAMETERS: p_custno TYPE s_customer.
SELECTION-SCREEN END OF SCREEN 0001.
```

e) From the menu bar, choose *Go To* → *Text Elements* → *Selection Texts*.

f) Select the *Dictionary Ref.* value provided in the table above and choose *Enter*.

g) Activate the text changes and go back to your program.

h) Save and activate the program.

7. To change the single index table from the standard, assign it to your interface in the namespace-specific features, and add your program and screen number for the selection.

Use the following data:

Field	Value
Interface	**IDOC_SCEN4**
Version	**1**
Message Index Table	**ZS###_IDX_4IDOC**
Program Name	**ZAIF_S###_4IDOC_SEL_SCREEN**
Screen Number	**0001**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Error Handling* → *Namespace-Specific Features* and enter your namespace.

b) In *Define namespace specific feature*, create a new entry.

c) Enter the data from the Define Namespace table.

d) Save.

8. To fill the interface-specific fields of your index table, the system needs the interface field providing them.

Note:
Use the interface-specific features to add the necessary information.

Use the following data:

Field	Value
Key Field	**SCUSTNO**
Data Element	**S_CUSTOMER**
Select-Options / Parameter	**P_CUSTNO** (use the parameter name created in the selection screen)
Field name	**E1SCU_CHG-CUSTOMERNUMBER**
Icon	**@A0@**
Tooltip	**Customer**

a) Navigate to AIF Customizing (transaction /AIF/CUST) under *Error Handling → Interface-Specific Features*.

b) Enter your namespace, interface, and version.

c) In Define Key Fields for Multi. Search, create a new entry. Enter sequence number 1.

d) Maintain the values in the Define Key Fields table.

e) Save.

9. Optional: Maintain a recipient.

If your user is still assigned to the ALL_INTERFACES recipient, you can skip this step. If your user is not assigned to the recipient, you must add it again.

a) Navigate to *System Configuration → Assign Recipients*.

b) Assign your user to the recipient.

10. To solve errors in Monitoring and Error Handling, create a corresponding editable field.

Use the following data:

Index: **10**

Field Path: **E1SCU_CHG-CUSTOMERNUMBER**

a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Error Handling → Interface-Specific Features*.

b) Choose your namespace, interface, and version.

c) Choose *Define Changeable Fields* and create a new entry.

d) Enter the values provided in the step for *Index* and *Field Path*.

e) Save.

Task 2: Test Your Development

1. Test your development by creating a new IDoc via transaction `WE19` and changing the customer.

> Note:
> Ensure you fill the corresponding X-Structure. Send a second IDoc using a non-existing customer number.

Use the following data:

Field	Value
Segment	E1SCU_CHG
Basic Type	ZAIF_S###_FLCUST_CHANGE01
EDIDC fields	
Port	**SAP<SYS-ID>** (replace SYS-ID with your system ID) for example, SAPT41
Partner No. Sender	**ZAIF_S###**
Part. Type Sender	**LS**
Partner No. Receiver	**ZAIF_S###**
Part. Type Receiver	**LS**
Message Type	**FLIGHTCUSTOMER_CHANGE**
E1BPSCUNEW fields	
Custname	**Fridolin Hirsch**
Street	**Bachstelzenweg 1**
Postcode	**47111**
City	**Changed city**

a) In transaction `WE19`, choose *Basic Type*, enter your basic type provided in the table above, and execute.
An empty IDoc structure displays.

b) Double-click on *EDIDC*, choose the *All Fields* button, and maintain the information for receiver, sender, and message type. Use the All EDIDC Fields table.

c) Close the dialog box and expand the node *E1SCU_CHG*. Enter your customer number in the field *CUSTOMERNUMBER*.

d) Close the dialog box and double-click on node *E1BPSCUNEW*. A dialog box appears. Maintain the data you want to change, for example *Custname*, *Street, Postcode*, and *City*. Use the data from the *E1BPSCUNEW* Fields table.

e) Close the dialog box and double-click on node *E1BPSCUNEX*. A dialog box appears. Maintain the fields corresponding to the fields you filled above with an X to tell the system you want to change them.

f) Close the dialog box and choose *Standard inbound* from the application toolbar. An IDoc is created and sent. The IDoc was processed with AIF and your user is assigned to a corresponding recipient. The IDoc will be displayed in the Interface Monitor.

g) Start the Interface Monitor via transaction `/AIF/IFMON`. On the right side, expand your namespace.
Your *IDOC_SCEN4* interface displays as a sub node. If everything works, the status icon is green. If you tried to change the customer with fields that contain errors (for example wrong customer type), the status icon is red.

h) Choose the *Sum* icon for all messages.
Monitoring and Error Handling starts. Monitoring and Error Handling is accessed from the Interface Monitor. Only the selected messages are displayed.

i) Choose the message node.
The application log messages display in the Application Log view. An error message indicates if something is wrong. If the message is posted successfully no error message occurs.

j) The structure of the interface displays in data. Choose the structure *E1BPSCUNEW*. The data transferred displays in *Data Content*. The structure contains the same data as entered in transaction `WE19`.

> Note:
> If you try to change the CUSTTYPE to a disallowed value (for example Z), you can change it here and repost the message. Scroll to the right and choose the *CUSTTYPE* field. Double-click the field. A dialog box appears. Enter **B** and close the dialog box. Save the data content.

k) In *Data Messages*, choose *Restart*. Choose the *Read* button.
After you resolve the error and reprocess the message, the icon in *Data Messages* changes to green (tooltip: *Application document posted*). A message displays that the customer is created in *Log Messages*.

l) Note the customer number and, in transaction `BC_GLOBAL_SCUST_DISP`, enter the customer number. Choose *Display*.
The customer you just created displays.

Unit 9
Exercise 19

Use a Customer-Specific Inbound Proxy Interface

Business Scenario

With the SAP Application Interface Framework, you can monitor and implement your own customer-specific proxy interfaces. Without AIF, you would usually implement the proxy's method with some custom-specific code to execute your business logic. With AIF, you simply call AIF and pass the data received by the proxy to AIF. The logic, for example, mapping and checks, is moved to AIF.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Note:
This exercise depends on completing the exercise Create a Value Mapping.

Task 1: Prepare the Interface

Before we can start with implementing an AIF proxy interface, we need a service interface. Create your own service interface in transaction `SE80`. Implement the proxy class method to call the AIF method `/AIF/CL_ENABLER_PROXY=>PROCESS_MESSAGE`.

1. Create a service interface.
   Use the following data:

Field	Value
Input field 1	**`Package as object Type`**
Input field 2	**`ZAIF_S###`**
Interface Details	
Service interface name	**`FlightBooking_in01`**
Namespace	**`http://www.aif.com/MDR/S###`**

2. Add operations to the service interface.
   Use the following data:

Context Menu Option	Namespace	Message Type
Name	Post_Bookings01	

Context Menu Option	Namespace	Message Type
Set Request → Select Existing Message Type	**http://www.aif.com/MDR/TRAINING**	**FlightBooking_MT**
Add Fault → Select Existing Messages Type	**http://www.aif.com/MDR/TRAINING**	**FlightBooking_Fault**

3. Create a proxy class implementation.
   Use the following data:

Field	Value
Implementing Class	ZAIF_S###_CL_FLIGHT_BOOKING_IN
Method	ZAIF_S###_II_FLIGHT_BOOKING_IN~POST_BOOKINGS01
Class Method	
Class	**/AIF/CL_ENABLER_PROXY**
Method	**PROCESS_MESSAGE**

Task 2: Create and Implement an AIF Interface
You want to use the server proxy you created as an inbound interface. Use your implemented class ZAIF_S###_CL_FLIGHT_BOOKING_IN as Proxy Class Inbound and /AIF/TEST_SAP_FLBOOKING as the SAP structure. The direction is inbound.

1. Define an interface.
   Use the following data:

Field	Value
Namespace	S###
Interface	
Interface name	**FLBOOK_IN**
Interface version	**1**
Proxy class	ZAIF_S###_CL_FLIGHT_BOOKING_IN
Interface direction	I (Inbound)

2. Check the correct engines are set for a proxy interface in AIF.
3. Map the source structure containing the booking data to the corresponding destination structure.
   Use the following data:

Field	Value
Source Structure	BOOKING_DATA
Assign Destination Structure	

Field	Value
Number of Structure Mappings	**10**
Destination Structure	**BOOKING_DATA**  Note: Select the value from F4 help.

4. Add a data element for conversion or a conversion routine.
   Add the mappings for the following fields:

Field in Destination Structure	Field Name 1
CUSTOMERID	CUSTOMER_ID
CLASS	CLASS
AGENCYNUM	AGENCY
PASSNAME	CUSTOMER_NAME
FLIGHTDATE	FLDATE
Data Element for Conversion	DATS
Direction of Conversion exit	External –> Internal

5. Add value mappings for fields that cannot be directly mapped.

 Note: Check if you can reuse value mappings from previous exercises.

The source structure does not contain the *AIRLINEID* and *CONNCTIONID* fields. Derive them from settings for the originating airport, the destination airport, and the flight date.

Use the following data:

Field	Value
AIRLINE ID	
Field in Destination Structure	**AIRLINEID**
Field Name 1	**AIRPORT_FROM**
Field Name 2	**AIRPORT_TO**

Field	Value
Field Name 3	**@FLIGHTDATE**  Note: To select this value from *Destination Structure*, choose F4 help. Choose *Switch Structure*.
Namespace	S###
Value Mapping	**VM_AIRLINEID**
CONNECTION ID	
Field in Destination Structure	**CONNECTID**
Field Name 1	**AIRPORT_FROM**
Field Name 2	**AIRPORT_TO**
Field Name 3	**@FLIGHTDATE**  Note: To select this value from *Destination Structure*, choose F4 help. Choose *Switch Structure*.
Field Name 4	**@AIRLINEID**
Namespace	S###
Value Mapping	**VM_CONNECTID**

6. Assign an action.

   To post the flights, an action has to be assigned. In namespace BIT750, there is an action FLBOOKING_CREATE, which can be reused. Ensure that you pass the correct record type.

   Use the following data:

Field	Value
Action Number	**10**
Namespace	**BIT750**

Field	Value
Action	**FLBOOKING_CREATE**  Note: To choose this value, use the F4 help of the field.

Task 3: Test the Interface

To test the processing of your interface, use the test environment of transaction *SE80*.

1. Test your interface.

   Use the following data:

Field	Value
Input field 1	**Package**
Input field 2	**Package ZAIF_S###**
Service provider	INTFZAIF_S###_II_FLIGHT_BOOKING_IN

2. In transaction `/AIF/ERR`, check if your message is received.

 Note:
Messages created in *SPROXY* do not persist. It is impossible to display the content or perform actions like editing, restarting, and canceling.

Unit 9
Solution 19

Use a Customer-Specific Inbound Proxy Interface

Business Scenario

With the SAP Application Interface Framework, you can monitor and implement your own customer-specific proxy interfaces. Without AIF, you would usually implement the proxy's method with some custom-specific code to execute your business logic. With AIF, you simply call AIF and pass the data received by the proxy to AIF. The logic, for example, mapping and checks, is moved to AIF.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Note:
This exercise depends on completing the exercise **Create a Value Mapping**.

Task 1: Prepare the Interface

Before we can start with implementing an AIF proxy interface, we need a service interface. Create your own service interface in transaction `SE80`. Implement the proxy class method to call the AIF method `/AIF/CL_ENABLER_PROXY=>PROCESS_MESSAGE`.

1. Create a service interface.
   Use the following data:

Field	Value
Input field 1	**`Package as object Type`**
Input field 2	**`ZAIF_S###`**
Interface Details	
Service interface name	**`FlightBooking_in01`**
Namespace	**`http://www.aif.com/MDR/S###`**

   a) Execute transaction `SE80`.

   b) In the two input fields, enter the first two data from the table above.

   c) In the *ZAIF_S###* node, in the context menu, choose *Create*.

d) In the opened list, choose *Enterprise Service*.
A wizard appears.

e) Select *Service Provider* and choose *Continue*.

f) Select *Backend* and continue.

g) On the next screen, enter the data provided in the table above, below **Interface Details**.

h) Choose *Continue*.

i) Enter your package *ZAIF_S###*, select your workbench request, and enter a prefix *ZAIF_S###_*.

j) Choose *Complete* and create the service provider.

2. Add operations to the service interface.
Use the following data:

Context Menu Option	Namespace	Message Type
Name	Post_Bookings01	
Set Request → Select Existing Message Type	**http://www.aif.com/MDR/TRAINING**	**FlightBooking_MT**
Add Fault → Select Existing Messages Type	**http://www.aif.com/MDR/TRAINING**	**FlightBooking_Fault**

a) Your service provider displays and you must add an operation.

b) Choose the *Internal View* tab.

c) In the *ZAIF_S###_II_FLIGHT_BOOKING_IN* context menu, choose *Add Operation*.

d) Enter the name provided in the step.

e) Using the data provided in the Message Types table, in your operation's context menu, enter the corresponding message type for each namespace.

f) Save and activate your service interface.

3. Create a proxy class implementation.
Use the following data:

Field	Value
Implementing Class	ZAIF_S###_CL_FLIGHT_BOOKING_IN
Method	ZAIF_S###_II_FLIGHT_BOOKING_IN~POST_BOOKINGS01
Class Method	
Class	**/AIF/CL_ENABLER_PROXY**
Method	**PROCESS_MESSAGE**

a) From your service provider, choose *Properties* and double click the *Implementing Class* provided in the step.

b) Double click on the method provided in the step.
You are in display mode.

c) Switch to *Change*.

d) Under the comment ****insert implementation here****, add your coding.

e) Choose *Pattern* and navigate to *ABAP Objects*.

f) Enter the data from the Class Method table.

g) Pass the *INPUT* from the proxy method to *IS_INPUT*.

h) Pass the name of your exception class to parameter *IV_EXCEPTION_CLASSNAME* of method *PROCESS_MESSAGE*.

Note:
You can get the exceptions name from the proxy method's signature.

You used an already defined fault message type. The exception's name is *ZAIF_T100_CX_FLIGHT_BOOKING_FA*.

Note:
When inserting call of method *PROCESS_MESSAGE* through the pattern, button, a try/catch block may be inserted depending on your user settings . However, do not catch an exception here. If the exception is caught here, an error is not passed to the proxy framework, the message is not in an error state, and it cannot be restarted.

The method implementation appears as in the following code:

```
METHOD
zaif_s###_ii_flight_booking_in~post_bookings01.

CALL METHOD /aif/cl_enabler_proxy=>process_message
  EXPORTING
    is_input               = input
    iv_exception_classname =
'ZAIF_T100_CX_FLIGHT_BOOKING_FA'.

ENDMETHOD.
```

i) Save.

j) Return to your implementation class and activate.
A dialog appears showing all items that the system is set to activate.

Note:
Choose all if some are not flagged by default.

k) Choose *Execute*.

Note:
If the dialog does not appear, discuss it with the instructor.

Task 2: Create and Implement an AIF Interface

You want to use the server proxy you created as an inbound interface. Use your implemented class ZAIF_S###_CL_FLIGHT_BOOKING_IN as Proxy Class Inbound and /AIF/TEST_SAP_FLBOOKING as the SAP structure. The direction is inbound.

1. Define an interface.

   Use the following data:

Field	Value
Namespace	S###
Interface	
Interface name	**FLBOOK_IN**
Interface version	**1**
Proxy class	ZAIF_S###_CL_FLIGHT_BOOKING_IN
Interface direction	I (Inbound)

   a) Execute transaction /AIF/CUST and create a new interface in *Interface Development → Define Interfaces*.

   b) Enter the namespace provided in the step and choose *New Entries*.

   c) Enter the interface data provided in the step and enter a description.

   Use the existing structure `/AIF/TEST_SAP_FLBOOKING` as destination structure. You are creating an inbound interface. As a result, the destination structure is the SAP data structure.

   d) In *Proxy Class Inbound*, enter the proxy class value provided in the step and press enter.

   The raw data structure `ZAIF_T100_FLIGHT_BOOKING_MT` and record type in raw structure `FLIGHT_BOOKING_MT` fill automatically. You create an inbound interface. As a result, the raw data structure is the source structure.

   e) Choose the value of the interface direction provided in the step.

   f) Save your interface.

2. Check the correct engines are set for a proxy interface in AIF.

   a) Navigate to *Additional Interface Properties → Specify Interface Engines* and display the data for your interface.

   The Application Engine and Persistence Engine must be set to *New Web Service*. AIF selects the persisted data from the proxy framework and the message also restarts in the proxy framework.

Note:
If you use PI as a middleware or you have NetWeaver 7.31 or lower, you must choose **`Proxy`** for the Application Engine and Persistence Engine.

The Selection Engine must be set to AIF Index Tables and the Logging Engine to AIF Application Log. These engines are the default ones. As a result, you need not change anything.

b) Save.

The SAP and the RAW structure are different. You must map a structure.

3. Map the source structure containing the booking data to the corresponding destination structure.

Use the following data:

Field	Value
Source Structure	BOOKING_DATA
Assign Destination Structure	
Number of Structure Mappings	**`10`**
Destination Structure	**`BOOKING_DATA`**  Note: Select the value from F4 help.

a) To create a structure mapping, navigate to *Interface Development* → *Define Structure Mapping*.

b) In *Select Source Structure*, choose the value for *New Entries* provided in the step. Don't forget to save.

Note:
Select the value from F4 help.

c) Mark the newly created entry and double click *Assign Destination* Structure.

d) Choose *New Entries* and enter the data from the table above, below **Assign Destination** structure .

4. Add a data element for conversion or a conversion routine.

Add the mappings for the following fields:

Field in Destination Structure	Field Name 1
CUSTOMERID	CUSTOMER_ID
CLASS	CLASS

Field in Destination Structure	Field Name 1
AGENCYNUM	AGENCY
PASSNAME	CUSTOMER_NAME
FLIGHTDATE	FLDATE
Data Element for Conversion	DATS
Direction of Conversion exit	External –> Internal

a) Navigate to *Define Field Mappings* and choose *New Entries*.

b) Using the data provided in the table above, create the field mappings.
If you have created one field mapping, choose *Next Entry* to continue.

In F4 help, the *FLIGHTDATE* is of type *DATS* and *FLDATE* is of type *STRING*.

So you need an additional Data Conversion for this field. Add a Data Element for conversion.

c) Enter the value of *Data Element for Conversion* provided in the step.

5. Add value mappings for fields that cannot be directly mapped.

Note:
Check if you can reuse value mappings from previous exercises.

The source structure does not contain the *AIRLINEID* and *CONNCTIONID* fields. Derive them from settings for the originating airport, the destination airport, and the flight date.

Use the following data:

Field	Value
AIRLINE ID	
Field in Destination Structure	**AIRLINEID**
Field Name 1	**AIRPORT_FROM**
Field Name 2	**AIRPORT_TO**
Field Name 3	**@FLIGHTDATE**  Note: To select this value from *Destination Structure*, choose F4 help. Choose *Switch Structure*.
Namespace	S###
Value Mapping	**VM_AIRLINEID**
CONNECTION ID	

Field	Value
Field in Destination Structure	**CONNECTID**
Field Name 1	**AIRPORT_FROM**
Field Name 2	**AIRPORT_TO**
Field Name 3	**@FLIGHTDATE**  Note: To select this value from *Destination Structure*, choose F4 help. Choose *Switch Structure*.
Field Name 4	**@AIRLINEID**
Namespace	S###
Value Mapping	**VM_CONNECTID**

a) Create new entries for the desired fields and use your Value Mappings for the *AIRLINEID* and the *CONNECTID* .

b) Enter the data provided in the table above below AIRLINE ID .

The Airline ID is not in the source structure but we can derive it. It is derived from the fields *AIRPORT_FROM, AIRPORT_TO*, and *FLIGHTDATE*.

c) Enter the data provided in the table above below Connection ID.

The Connection ID is not in the source structure but we can derive it. It is derived from the fields *AIRPORT_FROM, AIRPORT_TO, FLIGHTDATE*, and *AIRLINE ID*.

d) Save.

6. Assign an action.

To post the flights, an action has to be assigned. In namespace BIT750, there is an action FLBOOKING_CREATE, which can be reused. Ensure that you pass the correct record type.

Use the following data:

Field	Value
Action Number	**10**
Namespace	**BIT750**
Action	**FLBOOKING_CREATE**  Note: To choose this value, use the F4 help of the field.

a) In your structure mapping, navigate to *Assign Actions* and choose *New Entries*.

b) Enter the data provided in the New Action table.

c) Doubly click on the action.
The action displays.

d) Check the *Main Component Type*.
The main component type is `/BIT750/TEST_SAP_FLBOOKING`. This is your interface's SAP data structure type. The *Record Type* in *Assign Actions* remains empty because the action executes for the root structure.

e) Save your structure mapping.

Task 3: Test the Interface

To test the processing of your interface, use the test environment of transaction *SE80*.

1. Test your interface.
   Use the following data:

Field	Value
Input field 1	**`Package`**
Input field 2	**`Package ZAIF_S###`**
Service provider	INTFZAIF_S###_II_FLIGHT_BOOKING_IN

a) Execute transaction `SE80`.

b) Fill the input fields with the data provided in the Transaction Details table.

c) Open the *Nodes Enterprise Services* and *Service Providers*.

d) Double click on the service provider provided in the step.

e) Select *Test* (or press F8).

f) Choose *Test* and choose *Execute*.
An XML file displays with some generated data.

g) Choose *XML Editor* and change its contents so that it contains some flight booking data.

h) Choose *Execute*.

2. In transaction `/AIF/ERR`, check if your message is received.

Note:
Messages created in *SPROXY* do not persist. It is impossible to display the content or perform actions like editing, restarting, and canceling.

a) Execute transaction `/AIF/ERR`.

b) Fill the search screen with the data of your interface.
Ensure that you choose *Select All* so that messages can be seen successfully.
You see an entry in *Data Messages* and the log messages display.

Unit 9
Exercise 20

Create and Use an Outbound Proxy Interface

Business Scenario

In this exercise, you will create and use a proxy interface.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create a Service Interface

Before you can start with implementing an AIF proxy interface, you need a service interface.

1. Create your own service interface in transaction SE80.
   Use the following data:

Field	Value
Object Type	**ZAIF_S###**
Your package	**ZAIF_S###**
Wizard entry	Service Consumer
Service Interface	
Service interface name	**FlightBooking_Out01**
MDR namespace	**http://www.aif.com/MDR/S###**
Transport	
Your package	**ZAIF_S###**
Request/Task	Your Transport Request
Prefix	**ZAIF_S###_**
Operation	
Operation name	Post_Bookings01
Message type	**FlightBooking_MT**
Namespace	**http://www.aif.com/MDR/TRAINING**

Task 2: Create an AIF Interface

Monitoring and Error Handling displays the content of the source structure. However, in case of outbound interfaces, the data of the source structure is not persisted anywhere. Therefore, AIF has to take care of persisting this data. In order to enable AIF to do so, the application and persistence engines have to be set to the AIF's own persistence.

Create a structure mapping for your outbound interface. Map the source structure containing the flight information to the corresponding data of the destination structure.

Add a value mapping to those fields that cannot be directly derived from the source structure.

The destination structure has a field for the customer name. The customer's name is not contained in the source structure. However, the customer ID can be used to derive the name from table SCUSTOM with a value mapping. Create the field Mapping CUSTOMER NAME to do so.

Ensure that no data from an incorrect agency passes to the receiver. If the mapping fails, value mappings can send a message. They can check if you find more than one entry. You send the only key field, so you can add Message 011 of message class ZAIF_TRAINING to receive an error if the mapping fails.

1. Define the interface.

   Use the following data:

Field	Value
Namespace	S###
New interface name	FLBOOK_OUT
Interface version	1
Proxy class outbound	ZAIF_S###_CO_FLIGHT_BOOKING_OU
SAP Data Structure	ZAIF_T100_SAP_FLBOOKING_SEND

2. Set the application and persistence Engine to the AIF XML engine.

3. Create a structure mapping.

   Use the following data:

Field	Value
New Entries	FLIGHT_BOOKING
Number of Structure Mapping	10
Destination Structure	BOOKING_DATA

4. Map fields.

   Use the following data:

Field in Destination Structure	Field Name 1
AGENCY	**AGENCYNUM**
AIRPORT_FROM	**AIRPORT_FROM**
AIRPORT_TO	**AIRPORT_TO**
CUSTOMER_ID	**CUSTOMERID**
CLASS	**FLIGHTCLASS**
FLDATE	**FLDATE**

5. Add fields that cannot be directly derived from the source structure.

Use the following data:

Field	Value
Destination Structure	CUSTOMER_NAME
Field Name 1	CUSTOMERID
Namespace	S###
Value Mapping	GET_CUST_NAME
Description	Find Customer Name
Type ID	$1
Table Name	SCUSTOM
Field Name	NAME

6. Ensure that no data from an incorrect agency passes to the receiver.

   Use the following data:

Field	Value
Error message class	**ZAIF_TRAINING**
Error message number  Note: Use F4 help.	**011**
Field message variable 1 definition	**$1**  Note: This value links it to the first (and only) field you hand over to this value mapping.

Task 3: Create a Report to Send Messages

Triggering the sending of a message requires modules such as a report or function module. For this exercise, you use a simple report in which a user can enter some flight data (such as agency number, customer ID, airport from, airport to, flight date, and flight class). You can copy the `ZAIF_T100_SEND_FLBOOKING_TEMPL` report, which provides a template that already contains the parameters.

The template already provides the selection screen and the movement from input data to the interface's structure. Pass the filled structure to AIF by calling the `/AIF/SEND_WITH_PROXY` function module. Ensure you identify your interface correctly and pass the filled source structure.

1. Create the report.

   Use the following data:

Field	Value
Program	ZAIF_T100_SEND_FLBOOKING_TEMPL
Target	ZAIF_S###SEND_FLBOOKING
Package	ZAIF_S###

2. Pass the filled structure to AIF.
   Use the following data:

```
*&---------------------------------------------------------------------*
*
*& Report  ZAIF_T100_SEND_FLBOOKING_TEMPL
*&
*&---------------------------------------------------------------------*
*
*&
*&
*&---------------------------------------------------------------------*
*

REPORT  ZAIF_S###_SEND_FLBOOKING.

DATA: ls_booking TYPE zaif_t100_sap_flbooking_send.

PARAMETERS: p_agen    TYPE s_agncynum,
            p_custid  TYPE s_Customer,
            p_airfr   TYPE s_fromairp,
            p_airto   TYPE s_toairp,
            p_date    TYPE char10,
            p_class   TYPE s_class.

*1.) Add data to SAP Structure

ls_booking-flight_booking-airport_from = p_airfr.
ls_booking-flight_booking-airport_to = p_airto.
ls_booking-flight_booking-fldate = p_date.
ls_booking-flight_booking-customerid = p_custid.
ls_booking-flight_booking-agencynum = p_agen.
ls_booking-flight_booking-flightclass = p_class.

*2.) Call AIF

*Insert your code here
CALL FUNCTION '/AIF/SEND_WITH_PROXY'
 EXPORTING
   NS                          = 'S###'
   IFNAME                      = 'FLBOOK_OUT'
   IFVERSION                   = '1'
*   APPL_LOG_FUNCTION           = 'AIF_SEND_WITH_PROXY'
*   DO_COMMIT                   = 'X'
*   RESTART_MSG_GUID            =
*   MSGGUID                     =
*   IS_RESTART                  =
*   IV_PERSIST_XML              = 'X'
*   LOGICAL_PORT_NAME           =
*   IV_EOIO                     =
*   IV_QUEUE_ID                 =
* IMPORTING
*   XIMSGGUID                   =
*   XIMSGGUID_OUT               =
```

```
*TABLES
*  ADD_RETURN_TAB              =
 CHANGING
*   RESP_SAP_STRUCT            =
   SAP_STRUCT                  = ls_booking
*   ERROR_STRING               =
*   ERROR_LONG_TEXT            =
EXCEPTIONS
   PERSISTENCY_ERROR           = 1
   STATUS_UPDATE_FAILED        = 2
   MISSING_KEYS                = 3
   INTERFACE_NOT_FOUND         = 4
   TRANSFORMATION_ERROR        = 5
   GENERAL_ERROR               = 6
   OTHERS                      = 7
        .
IF SY-SUBRC <> 0.
Write: ' Error in Function Module ' .
ENDIF.
```

Task 4: Test the Implementation

1. Execute your report.

   Use the following data:

Field	Value
Report name	ZAIF_S###_SEND_FLBOOKING

2. Test your development.

   Use the following data:

Field	Value
First Customer	
Customer ID	7
Second Customer	
Customer ID	99999

Unit 9
Solution 20

Create and Use an Outbound Proxy Interface

Business Scenario

In this exercise, you will create and use a proxy interface.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create a Service Interface

Before you can start with implementing an AIF proxy interface, you need a service interface.

1. Create your own service interface in transaction SE80.
   Use the following data:

Field	Value
Object Type	**ZAIF_S###**
Your package	**ZAIF_S###**
Wizard entry	Service Consumer
Service Interface	
Service interface name	**FlightBooking_Out01**
MDR namespace	**http://www.aif.com/MDR/S###**
Transport	
Your package	**ZAIF_S###**
Request/Task	Your Transport Request
Prefix	**ZAIF_S###_**
Operation	
Operation name	Post_Bookings01
Message type	**FlightBooking_MT**
Namespace	**http://www.aif.com/MDR/TRAINING**

   a) In your training system, execute transaction `SE80`.

   b) In the two input fields, enter the data provided in the table above.

c) Press *Return*. Your package displays.

d) Right-click on the *ZAIF_S###* node and choose *Create*.

e) Choose enterprise service.
A wizard opens.

f) In the wizard, select the value provided in the step and choose *Continue*.

g) Select *Backend* and continue.

h) On the next screen, enter the data provided in the **Transport** part of the table above, and choose *Continue*.

i) Enter the data from the ZAIF table and choose *Continue*.

j) On the next screen, choose *Complete*.
Your service consumer displays.

k) To add an operation, in *Internal View*, right-click on *ZAIF_S###_CO_FLIGHT_BOOKING_OU* and select *Add Operation*. Enter the name provided in the **Operation** part of the table above. Hit *Enter*.

l) Right click on your operation and choose *Set Request → Select Existing Message Type*. Search for the data provided in the Message Type table. choose it the message type.

m) Move to the *External View* tab.

n) In the *External View* tab, in the *Field Mapping Mode* tab, choose *Coupled*.

o) Save and activate your service interface.

Note:
In the unlikely case, that your user is not authorized to activate the service, ask your trainer.

Task 2: Create an AIF Interface

Monitoring and Error Handling displays the content of the source structure. However, in case of outbound interfaces, the data of the source structure is not persisted anywhere. Therefore, AIF has to take care of persisting this data. In order to enable AIF to do so, the application and persistence engines have to be set to the AIF's own persistence.

Create a structure mapping for your outbound interface. Map the source structure containing the flight information to the corresponding data of the destination structure.

Add a value mapping to those fields that cannot be directly derived from the source structure.

The destination structure has a field for the customer name. The customer's name is not contained in the source structure. However, the customer ID can be used to derive the name from table SCUSTOM with a value mapping. Create the field Mapping CUSTOMER NAME to do so.

Ensure that no data from an incorrect agency passes to the receiver. If the mapping fails, value mappings can send a message. They can check if you find more than one entry. You send the only key field, so you can add Message 011 of message class ZAIF_TRAINING to receive an error if the mapping fails.

1. Define the interface.

Use the following data:

Field	Value
Namespace	S###
New interface name	FLBOOK_OUT
Interface version	1
Proxy class outbound	ZAIF_S###_CO_FLIGHT_BOOKING_OU
SAP Data Structure	ZAIF_T100_SAP_FLBOOKING_SEND

a) Navigate to AIF Customizing transaction `/AIF/CUST`.

b) Choose *Interface Development* → *Define Interface*.

c) Choose the namespace provided in the step.

d) On the next screen, choose new entries.

e) Enter the new interface name and version provided in the step.

f) Add a description.

g) Enter the value for the proxy class outbound provided in the step and choose *Enter*.
The RAW data structure and the record Type in Raw structure is filled automatically.

h) Enter the value for the SAP Data Structure provided in the step.

Note:
If the fields are not filled automatically, check if the proxy class is active.

i) Save your interface.

If asked, choose your transport request.

2. Set the application and persistence Engine to the AIF XML engine.

a) In AIF Customizing, navigate to *Interface Development* → *Additional Interface Properties* → *Specify Interface Engines*.

b) On the popup, choose the namespace provided in the step.

c) Select your new outbound interface.

d) Set Application Engine and Persistence Engine as XML.

Leave the Selection Engine AIF Index Tables and Logging Engine AIF Application Log as they are.

e) Save.

f) If asked, choose your transport request.

3. Create a structure mapping.

Use the following data:

Field	Value
New Entries	FLIGHT_BOOKING

Field	Value
Number of Structure Mapping	10
Destination Structure	BOOKING_DATA

a) Navigate to *Interface Development → Define Structure Mappings* and select your outbound interface.

b) In *Select Source Structures*, choose the value for *New Entries* provided in the step from the F4 help.

c) Choose *Return*.

d) Select the created entry.

e) Navigate to *Assign Destination Structure*, click *New Entries*, and enter the values provided in the step for Number of Structure Mapping and Destination Structure.

4. Map fields.

   Use the following data:

Field in Destination Structure	Field Name 1
AGENCY	**AGENCYNUM**
AIRPORT_FROM	**AIRPORT_FROM**
AIRPORT_TO	**AIRPORT_TO**
CUSTOMER_ID	**CUSTOMERID**
CLASS	**FLIGHTCLASS**
FLDATE	**FLDATE**

a) Navigate to *Define Field Mapping*.

b) Choose new entries and create the simple 1:1 field mappings as displayed in the table.

You can move between the entries with F7 and F8.

c) Save.

If asked, use your transport request.

5. Add fields that cannot be directly derived from the source structure.

   Use the following data:

Field	Value
Destination Structure	CUSTOMER_NAME
Field Name 1	CUSTOMERID
Namespace	S###
Value Mapping	GET_CUST_NAME
Description	Find Customer Name
Type ID	$1

Field	Value
Table Name	SCUSTOM
Field Name	NAME

a) Navigate to *Define Field Mapping* and choose new entries.

b) In Create a new field mapping, enter the values provided in the step for Destination Structure and *Field Name 1*.

Note:
Use F4 Help.

c) Enter the values provided in the step for *Namespace* and Value Mapping and save.

Note:
If the system states the mapping does not exist, press return to continue.

d) In the dialog, choose *Yes* and choose your transport request.
The value mapping displays.

e) Choose *Change* to continue.

f) Enter the description provided in the step.

g) As *Table Name* enter **`SCUSTOM`**.

h) In *Field Name*, use F4 to choose name.

i) In *Where Condition for Select Statement*, enter **`ID = '$1'`**.
$1 refers to the first value that is handed over to this mapping.

6. Ensure that no data from an incorrect agency passes to the receiver.
Use the following data:

Field	Value
Error message class	**`ZAIF_TRAINING`**
Error message number  Note: Use F4 help.	**`011`**

Field	Value
Field message variable 1 definition	**$1**  Note: This value links it to the first (and only) field you hand over to this value mapping.

a) Enter the data provided in the table.

b) Save and choose your transport request.

Task 3: Create a Report to Send Messages

Triggering the sending of a message requires modules such as a report or function module. For this exercise, you use a simple report in which a user can enter some flight data (such as agency number, customer ID, airport from, airport to, flight date, and flight class). You can copy the `ZAIF_T100_SEND_FLBOOKING_TEMPL` report, which provides a template that already contains the parameters.

The template already provides the selection screen and the movement from input data to the interface's structure. Pass the filled structure to AIF by calling the `/AIF/SEND_WITH_PROXY` function module. Ensure you identify your interface correctly and pass the filled source structure.

1. Create the report.

   Use the following data:

Field	Value
Program	ZAIF_T100_SEND_FLBOOKING_TEMPL
Target	ZAIF_S###SEND_FLBOOKING
Package	ZAIF_S###

a) Navigate to transaction `SE38`.

b) In the field *Program*, enter the value provided in the step.

c) Choose *Copy*.

d) In the field *Target*, enter the value provided in the step and choose copy.

e) In the dialog that appears, choose copy again.
Another dialog appears. You are back on the SE38 screen with your new program filled in the *Program* field.

 Note:
If your package is not filled automatically, use the value provided in the step and choose *Save*. Use your transport request.

f) To edit your program, choose *Change*.

2. Pass the filled structure to AIF.

   Use the following data:

```
*&---------------------------------------------------------------------*
*
*& Report  ZAIF_T100_SEND_FLBOOKING_TEMPL
*&
*&---------------------------------------------------------------------*
*
*&
*&
*&---------------------------------------------------------------------*
*

REPORT  ZAIF_S###_SEND_FLBOOKING.

DATA: ls_booking TYPE zaif_t100_sap_flbooking_send.

PARAMETERS: p_agen    TYPE s_agncynum,
            p_custid  TYPE s_Customer,
            p_airfr   TYPE s_fromairp,
            p_airto   TYPE s_toairp,
            p_date    TYPE char10,
            p_class   TYPE s_class.

*1.) Add data to SAP Structure

ls_booking-flight_booking-airport_from = p_airfr.
ls_booking-flight_booking-airport_to = p_airto.
ls_booking-flight_booking-fldate = p_date.
ls_booking-flight_booking-customerid = p_custid.
ls_booking-flight_booking-agencynum = p_agen.
ls_booking-flight_booking-flightclass = p_class.

*2.) Call AIF

*Insert your code here
CALL FUNCTION '/AIF/SEND_WITH_PROXY'
 EXPORTING
   NS                          = 'S###'
   IFNAME                      = 'FLBOOK_OUT'
   IFVERSION                   = '1'
*   APPL_LOG_FUNCTION           = 'AIF_SEND_WITH_PROXY'
*   DO_COMMIT                   = 'X'
*   RESTART_MSG_GUID            =
*   MSGGUID                     =
*   IS_RESTART                  =
*   IV_PERSIST_XML              = 'X'
*   LOGICAL_PORT_NAME           =
*   IV_EOIO                     =
*   IV_QUEUE_ID                 =
* IMPORTING
*   XIMSGGUID                   =
*   XIMSGGUID_OUT               =
*TABLES
*  ADD_RETURN_TAB               =
 CHANGING
*   RESP_SAP_STRUCT             =
   SAP_STRUCT                  = ls_booking
*   ERROR_STRING                =
*   ERROR_LONG_TEXT             =
EXCEPTIONS
   PERSISTENCY_ERROR           = 1
```

```
   STATUS_UPDATE_FAILED        = 2
   MISSING_KEYS                = 3
   INTERFACE_NOT_FOUND         = 4
   TRANSFORMATION_ERROR        = 5
   GENERAL_ERROR               = 6
   OTHERS                      = 7
         .
IF SY-SUBRC <> 0.
Write: ' Error in Function Module ' .
ENDIF.
```

a) Add a new line after the comment `*insert your code here.`

b) To add the function module to the coding, use the Button Pattern.

c) Fill `/AIF/SEND_WITH_PROXY` in the field *CALL FUNCTION* and press Return.
This fills the function in the coding. All optional fields have the comment sign `*` in front of them.

d) Change the code to the data provided in the step.

e) Choose *Save* and your transport request.

f) Choose *Activate*.

g) In the next screen, select both lines and press return.

Note:
If there is a syntax error, ask your instructor for help.

Task 4: Test the Implementation

1. Execute your report.
   Use the following data:

Field	Value
Report name	ZAIF_S###_SEND_FLBOOKING

a) Execute transaction `SE38`.

b) In the *Program* field, fill in the report name provided in the table above.

c) Press `F8`.

2. Test your development.
   Use the following data:

Field	Value
First Customer	
Customer ID	7
Second Customer	

Field	Value
Customer ID	99999

a) Change the customer number to the first value provided in the table above. Choose *Execute*.

   The report runs successful.

b) Change the customer number to the second value provided in the table above. Choose *Execute*.
   You receive the message **`Error in function Module`** or **`Transformation failed`**.

c) Call the Monitoring and Error Handling transaction (`/AIF/ERR`).

d) Fill in the data of your interface in the selection screen.

e) To see the successful messages, choose the *Select All* button.

f) Choose *Execute*.
   You see one successful and one faulty message.

g) Double-click on the faulty message.
   You see the message **`Customer 99999 does not exist`**. You customized this message in the value mapping in case no entry is found in *SCUSTOM*.

Unit 9
Exercise 21

Link Inbound And Outbound Interface (Optional)

Business Scenario

This exercise is an optional exercise.

It is only suitable for participants familiar with transaction `SOAMANAGER`. It is not demonstrated during the training because transaction `SOAMANAGER` is not part of this training.

You created an inbound and an outbound interface. The outbound interface did not have a real partner. It is possible to create a webservice and a webservice client from the two interfaces and connect them to each other. This is done with transaction `SOAMANAGER`.

First, create the webservice for the inbound proxy, because it is needed in the definition of the outbound webservice. Second, create the outbound webservice and set it to use the WSDL from the inbound interface as a target service. After you create the settings in SOAMANAGER, pass the logical port to `/AIF/SEND_WITH_PROXY`.

Note:
This exercise depends on completing the following exercises:

- Use a Customer-Specific Inbound Proxy Interface
- Create and Use an Outbound Proxy Interface

Note:
To successfully perform this exercise, it is required, that the exercise **Create and Use an Outbound Proxy Interface** is completed.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Link Inbound and Outbound Interfaces

1. Create the webservice for the inbound proxy.

   Use the following data:

Field	Value
Your entry	

Field	Value
Name	FlightBooking_in01
Namespace	http://www.aif.com/MDR/S###
Service	
Service name	**FLBOOKING_S###**
New binding name	**FLBOOKING_S###**
Description	<of your choice>
Authentication	
Transport Level Security	**None (http)**
Authentication Settings	**No Authentication**

2. Create the outbound webservice and ensure it uses the WSDL from the inbound interface as target service.
3. Pass the logical port to `/AIF/SEND_WITH_PROXY`.

   Use the following data:

Field	Value
Program	Your program name
LOGICAL_PORT_NAME	'FLBOOKING_S###'

4. Test your webservices and interfaces with your report.

   Use the following data:

Field	Value
Executed report	data from the exercise, Create and Use an Outbound Proxy Interface
Interface name	***** [an asterisk]
Version	**1**

Unit 9
Solution 21

Link Inbound And Outbound Interface (Optional)

Business Scenario

This exercise is an optional exercise.

It is only suitable for participants familiar with transaction SOAMANAGER. It is not demonstrated during the training because transaction SOAMANAGER is not part of this training.

You created an inbound and an outbound interface. The outbound interface did not have a real partner. It is possible to create a webservice and a webservice client from the two interfaces and connect them to each other. This is done with transaction SOAMANAGER.

First, create the webservice for the inbound proxy, because it is needed in the definition of the outbound webservice. Second, create the outbound webservice and set it to use the WSDL from the inbound interface as a target service. After you create the settings in SOAMANAGER, pass the logical port to /AIF/SEND_WITH_PROXY.

Note:
This exercise depends on completing the following exercises:

- Use a Customer-Specific Inbound Proxy Interface
- Create and Use an Outbound Proxy Interface

Note:
To successfully perform this exercise, it is required, that the exercise **Create and Use an Outbound Proxy Interface** is completed.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Link Inbound and Outbound Interfaces

1. Create the webservice for the inbound proxy.

   Use the following data:

Field	Value
Your entry	

Field	Value
Name	FlightBooking_in01
Namespace	http://www.aif.com/MDR/S###
Service	
Service name	**FLBOOKING_S###**
New binding name	**FLBOOKING_S###**
Description	<of your choice>
Authentication	
Transport Level Security	**None (http)**
Authentication Settings	**No Authentication**

a) Execute transaction SOAMANAGER and choose *Web Service Configuration*.

b) Search for the service definition of your inbound service interface.
As *Object name* enter *Flight**.

c) Select your entry *FlightBooking_in01* in *Namespace* **http://www.aif.com/MDR/S###**.

d) In *Configurations*, choose *Create Service*.

e) Enter the data provided in the table above, below **Service**. Enter a description, and choose *Next*.

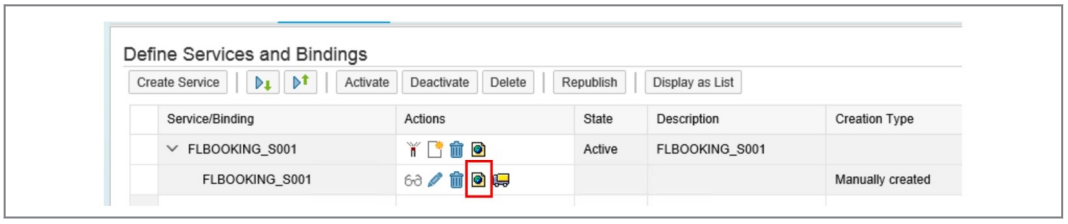
f) For *Authentication*, enter the data provided in the table above, below **Authentication**.

g) In the user name and password fields, enter your user and password.

h) Choose *Next*.

i) Choose *Finish*.

j) Select *Define Services and Bindings*



k) Select the button that resembles a webpage: .

l) In the dialog that appears, copy the *WSDL URL for Binding* to your clipboard.

m) Close the dialog and choose *Back*.

2. Create the outbound webservice and ensure it uses the WSDL from the inbound interface as target service.

a) Change to the search by consumer proxy. To do so, use the transaction `OWN BACK MENU`.

b) Select your consumer proxy: **`FlightBooking_Out01`** in the *Namespace* **`http://www.aif.com/MDR/S###`**.

Note:
Leave the *Flight** in the comparison field.

c) In *Configurations*, choose *Create*, and select **`WSDL based configuration`**. Enter **`FLBOOKING_S###`** in *Logical Port Name*.

d) Enter a description.

e) Choose *Next*.

f) Select *via HTTP Access:*

g) Paste the binding's URL that you copied before into *URL for WSDL Access*.

h) Enter your user and password in *WSDL Access User*.

i) Choose *Next*.
The overview of possible combinations Identified by WSDL Document displays.

j) Choose *Next* and, when prompted, enter your user and password.

k) Choose *Next* and choose *Finish*.

3. Pass the logical port to `/AIF/SEND_WITH_PROXY`.
Use the following data:

Field	Value
Program	Your program name
LOGICAL_PORT_NAME	'FLBOOKING_S###'

a) Execute transaction `SE38`.

b) Enter the value provided in the step for *Program* and choose *Change*.

c) In the call of the function module, delete `*` (the asterisk) in front of the *LOGICAL_PORT_NAME* line

d) Enter the value provided in the step for *LOGICAL_PORT_NAME*.

Note:
Ener this value after the = that follows *LOGICAL_PORT_NAME*.

e) Save and activate your report.

f) The report resembles the following code:

```
*&---------------------------------------------------------------------*
*
*& Report  ZAIF_T100_SEND_FLBOOKING_TEMPL
*&
*&---------------------------------------------------------------------*
*
*&
*&
*&---------------------------------------------------------------------*
*

REPORT  ZAIF_S###_SEND_FLBOOKING.

DATA: ls_booking TYPE zaif_t100_sap_flbooking_send.

PARAMETERS: p_agen    TYPE n LENGTH 8,"s_agncynum,
            p_custid  TYPE s_Customer,
            p_airfr   TYPE s_fromairp,
            p_airto   TYPE s_toairp,
            p_date    TYPE char10,
            p_class   TYPE s_class.

*1.) Add data to SAP Structure

ls_booking-flight_booking-airport_from = p_airfr.
ls_booking-flight_booking-airport_to = p_airto.
ls_booking-flight_booking-fldate = p_date.
ls_booking-flight_booking-customerid = p_custid.
ls_booking-flight_booking-agencynum = p_agen.
ls_booking-flight_booking-flightclass = p_class.

*2.) Call AIF

*Insert your code here
CALL FUNCTION '/AIF/SEND_WITH_PROXY'
 EXPORTING
   NS                          = 'S###'
   IFNAME                      = 'FLBOOK_OUT'
   IFVERSION                   = '1'
*   APPL_LOG_FUNCTION           = 'AIF_SEND_WITH_PROXY'
*   DO_COMMIT                   = 'X'
*   RESTART_MSG_GUID            =
*   MSGGUID                     =
*   IS_RESTART                  =
*   IV_PERSIST_XML              = 'X'
   LOGICAL_PORT_NAME           = 'FLBOOKING_S###'
*   IV_EOIO                     =
*   IV_QUEUE_ID                 =
* IMPORTING
*   XIMSGGUID                   =
*   XIMSGGUID_OUT               =
*TABLES
*  ADD_RETURN_TAB               =
 CHANGING
*   RESP_SAP_STRUCT             =
   SAP_STRUCT                  = ls_booking
*   ERROR_STRING                =
*   ERROR_LONG_TEXT             =
EXCEPTIONS
   PERSISTENCY_ERROR           = 1
   STATUS_UPDATE_FAILED        = 2
```

```
   MISSING_KEYS                  = 3
   INTERFACE_NOT_FOUND           = 4
   TRANSFORMATION_ERROR          = 5
   GENERAL_ERROR                 = 6
   OTHERS                        = 7
          .
IF SY-SUBRC <> 0.
Write: ' Error in Function Module ' .
ENDIF.
```

4. Test your webservices and interfaces with your report.

   Use the following data:

Field	Value
Executed report	data from the exercise, Create and Use an Outbound Proxy Interface
Interface name	* [an asterisk]
Version	1

   a) Execute transaction `SE38`.

   b) Enter your report name and choose *Execute*.

   c) Enter the required data from the exercise, Create and Use an Outbound Proxy Interface.
   If it is successful, there is no message.

   d) In transaction `/AIF/ERR`, enter the data from the table.

   e) Choose *Select All*.

   f) There is a list of interfaces with one node per interface.

   g) Open the nodes for *FLBOOK_IN* and *FLBOOK_OUT*.
   As the inbound and outbound Interfaces use the same *MessageID* and we use the Standard Single Index Table you will see only the inbound Interface.

Unit 10
Exercise 22

Use the File Adapter

Business Scenario

The SAP Application Interface Framework has a file adapter. You can use the file adapter to upload files to the SAP system and process the data in an AIF interface. The processing of the data can be monitored in the AIF interface.

In this exercise, you set up a simple file adapter scenario that creates new materials in the system.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Prepare the Exercise

Before you can start to implement the file adapter scenario, you require a CSV file containing the data you want to import into our system.

1. Create a CSV file.

   Use the following data:

Field	Value
Text document name	Material.csv

2. Add entries to the file.

   Use the following data:

Material	Text	MTyp	Industry	Unit
MAT_1	**CPU**	**HALB**	**M**	**EA**
MAT_2	**CPU**	**HALB**	**M**	**EA**
MAT_3	**Box**	**ROH**	**M**	**EA**
MAT_4	**Mouse**	**ROH**	**M**	**EA**
MAT_5	**Cable 2m**	**ROH**	**M**	**EA**
MAT_6	**Cable 1m**	**ROH**	**M**	**EA**

3. Create raw structure.

   Use the following data:

   Structure: **ZAIF_S###_MATERIAL_CSV**

Note:
The structure contains the fields used in the CSV file.

Component	Data Type	Length
Material	CHAR	18
Text	CHAR	70
Material_Type	CHAR	4
Industry	CHAR	1
UOM	CHAR	3

Note:
The typing method is always types.

4. Create a table type.
   Use the following data:

Field	Value
Structure	ZAIF_S###_MATERIAL_CSV
Table type	ZAIF_S###_MATERIAL_CSV_TT

5. Create a root structure.
   Use the following data:

Field	Value
Root structure	ZAIF_S###_RAW_MATERIAL
Table type	ZAIF_S###_MATERIAL_CSV_TT
Component	
Lines	ZAIF_S###_MATERIAL_CSV_TT (use the table type created above)

Task 2: Create the SAP Structure

You want to use the function module ‘BAPI_MATERIAL_SAVEDATA’ in the action to process the material data. You will fill HEADDATA, CLIENTDATA, CLIENTDATAX, with information to create a new material. Also, we want to include the material descriptions. The new SAP structure must be able to contain a table of materials. The materials should contain at least the structures named above.

Check ‘BAPI_MATERIAL_SAVEDATA’ to find out which components are required for the structure.

1. Create a structure (ZAIF_S###_MATERIAL) to be used as line type for the SAP structure. The structure contains one specific material.

Use the following data:

Field	Value
Headdata	**BAPIMATHEAD**
Clientdata	**BAPI_MARA**
Clientdatax	**BAPI_MARAX**
Materialdescription	**T_BAPI_MAKT**

Note:
The typing method is always types.

2. Create a table type.

Use the following data:

Field	Value
Data type	ZAIF_S###_Material_TT
Line Type	ZAIF_S###_MATERIAL

3. Create the SAP structure.

Use the following data:

Field	Value
Data type	ZAIF_S###_SAP_Material
Materials	ZAIF_S###_MATERIAL_TT (name of the table type created above)

Note:
The typing method is always types.

Task 3: Create an Interface

Create a new interface MAT_CREATE Version 1 in your Namespace S###. Use the newly created structures as SAP and RAW data structure.

1. Define an interface.

Use the following data:

Field	Value
Interface Name	MAT_CREATE
Interface Version	1
Description	Create Material via File

Field	Value
SAP Data Structure	ZAIF_S###_SAP_MATERIAL (use the name of the above created SAP structure)
Raw Data Structure	ZAIF_S###_RAW_MATERIAL (use the name of the above created raw structure)

2. Define interface engines.
   Use the following data:

Note:
The file adapter uses AIF's own XML persistence and runtime. Ensure that you correctly set the application and persistence engines.

Task 4: Define Mappings

1. Create structure mappings.
   Use the following data:

Field	Value
New entry	LINES
Number of Structure Mapping	10
Destination Structure	MATERIALS (select the table containing the materials in the SAP structure)

2. Create field mappings.
   Use the following data:

Field	Field Name 1	Field Name 2
CLIENTDATA-BASE_UOM	**UOM**	
HEADDATA-IND_SECTOR	**INDUSTRY**	
HEADDATA-MATERIAL	**%S###_**	**MATERIAL**
HEADDATA-MATL_TYPE	**MATERIALTYPE**	

3. Create fix values.
   Use the following data:

Field	Value
CLIENTDATAX-BASE_UOM	**X**
HEADDATA-BASIC_VIEW	**X**

Task 5: Define Indirect Mappings

Map the material description. The material descriptions are in the same table as the materials. For each material, you do not want all descriptions but only the correct one. You need an indirect mapping to connect them correctly.

1. Create another field mapping.

   Use the following data:

Field	Value
Field in Destination Structure	MATERIALDESCRIPTION
Sub-Table	LINES (the table containing the materials of the raw structure)
Selection Field	MATERIAL
Operator	EQ Equal
Comparison Field	MATERIAL

2. Create an indirect structure mapping for the sub-table.

   Use the following data:

Field	Value
Number of Structure Mapping	**20**
Destination Structure	**MATERIALDESCRIPTION**
Indirect Mapping	**X**

3. Define fix values.

   Use the following data:

Field	Value
Field Name	**LANGU**
Value	**E**

4. Define field mapping.

   Use the following data:

   Map the *MATL_DESC* to the *TEXT* field.

Field	Value
Field in Destination Structure	**MATL_DESC**
Field Name1	**TEXT**

Task 6: Define an Action

Using the forward navigation in Assign Actions, create a new action to process the files.

1. Create a new action.

   Use the following data:

Field	Value
Action Number	**10**
Namespace	**S###**
Action	**MATERIAL_CREATE**

Field	Value
Record Type	**MATERIALS**  Note: This table contains the materials in the SAP structure.

2. Add some data to the action.

   Use the following data:

Field	Value
Action description	Create material
Commit Mode	W Commit Work and Wait
Commit Level	F After Each Function
Main Component Type	**ZAIF_S###_MATERIAL** (The name of the structure containing the material in the SAP structure)

3. Create a function for your action.

   Use the following data:

Field	Value
Function Number	**10**
Function Module Name	**ZAIF_S###_AC_MATERIAL_CREATE**
Function group	ZAIF_S###
CURR_LINE	
parameter type	ZAIF_S###_MATERIAL

Code

```
DATA: ls_return TYPE bapiret2.

  CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
    EXPORTING
      headdata            = curr_line-headdata
      clientdata          = curr_line-clientdata
      clientdatax         = curr_line-clientdatax
    IMPORTING
      return              = ls_return
    TABLES
      materialdescription = curr_line-materialdescription.

  APPEND ls_return TO return_tab.
```

Task 7: Configure a File Adapter

You want to pass the material data to AIF via your created file. To read this file, configure the file adapter in AIF Customizing.

1. Configure the file adapter in AIF Customizing.

   Use the following data:

Field	Value
Config. ID	**MAT_CREATE**
File Type	**Text File**
File Content	**Flat Structure**
Text Type	**Special Character as Separator**
Field Separator	In order to get the correct separator open your txt file in notepad and check for the correct separator.
Header Line Exists	**X**
Structure Name	**ZAIF_S###_RAW_MATERIAL** (the name of the raw structure created above)
Field Name	**LINES**

Task 8: Test Your Development

You want to load files to AIF with your file adapter configuration.

1. Load files to AIF.

   Use the following data:

Field	Value
Config. Namespace	**S###**
Config. ID	**MAT_CREATE**

2. Check the result.

Unit 10
Solution 22

Use the File Adapter

Business Scenario

The SAP Application Interface Framework has a file adapter. You can use the file adapter to upload files to the SAP system and process the data in an AIF interface. The processing of the data can be monitored in the AIF interface.

In this exercise, you set up a simple file adapter scenario that creates new materials in the system.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Prepare the Exercise

Before you can start to implement the file adapter scenario, you require a CSV file containing the data you want to import into our system.

1. Create a CSV file.

   Use the following data:

Field	Value
Text document name	Material.csv

   a) On your local desktop, navigate to *Documents* (on Windows) or some equivalent location.

   b) Create a text document with the name provided in the step.

2. Add entries to the file.

   Use the following data:

Material	Text	MTyp	Industry	Unit
MAT_1	CPU	HALB	M	EA
MAT_2	CPU	HALB	M	EA
MAT_3	Box	ROH	M	EA
MAT_4	Mouse	ROH	M	EA
MAT_5	Cable 2m	ROH	M	EA
MAT_6	Cable 1m	ROH	M	EA

a) Open the text document.

b) Separating each value by a comma, enter the data from the table.

3. Create raw structure.
   Use the following data:

   Structure: **ZAIF_S###_MATERIAL_CSV**

> Note:
> The structure contains the fields used in the CSV file.

Component	Data Type	Length
Material	CHAR	18
Text	CHAR	70
Material_Type	CHAR	4
Industry	CHAR	1
UOM	CHAR	3

> Note:
> The typing method is always types.

a) Execute transaction `SE11`.

b) Choose *Data type* and, in the field beside it, enter the structure value provided in the step.

c) Choose *Create*.

d) In the dialog, choose *Structure*.

e) Enter a description.

f) Enter the data from the table.

g) To fill in the data type and length press the *Built-in Type* button.

h) Save and activate.
   Use your package and transport request. Ignore the Warning that an enhancement category is missing.

4. Create a table type.
   Use the following data:

Field	Value
Structure	ZAIF_S###_MATERIAL_CSV
Table type	ZAIF_S###_MATERIAL_CSV_TT

a) Choose *Data type* and, in the field beside it, enter the table type value provided in the step.

b) Choose *Create*.

c) In the dialog, choose *Table Type*.

d) Enter a description.

e) In the field beside the preselected *Line Type*, enter the structure value provided in the step.

f) Save and activate.

Use your package and transport request.

5. Create a root structure.

Use the following data:

Field	Value
Root structure	ZAIF_S###_RAW_MATERIAL
Table type	ZAIF_S###_MATERIAL_CSV_TT
Component	
Lines	ZAIF_S###_MATERIAL_CSV_TT (use the table type created above)

a) Execute transaction `SE11`.

b) Choose *Data type* and, in the field beside it, enter the root structure value provided in the step.

c) Choose *Create*.

d) In the dialog, choose *Structure*.

e) Enter a description.

f) Add the component as provided in the table above.

g) Save and activate.

Use your package and transport request. Ignore the Warning that an enhancement category is missing.

Task 2: Create the SAP Structure

You want to use the function module 'BAPI_MATERIAL_SAVEDATA' in the action to process the material data. You will fill HEADDATA, CLIENTDATA, CLIENTDATAX, with information to create a new material. Also, we want to include the material descriptions. The new SAP structure must be able to contain a table of materials. The materials should contain at least the structures named above.

Check 'BAPI_MATERIAL_SAVEDATA' to find out which components are required for the structure.

1. Create a structure (ZAIF_S###_MATERIAL) to be used as line type for the SAP structure. The structure contains one specific material.

Use the following data:

Field	Value
Headdata	**BAPIMATHEAD**
Clientdata	**BAPI_MARA**
Clientdatax	**BAPI_MARAX**
Materialdescription	**T_BAPI_MAKT**

Note:
The typing method is always types.

a) Execute transaction `SE11`.

b) Choose *Data type* and enter **ZAIF_S###_Material** in the field besides.

c) Choose *Create*.

d) In the dialog, choose *Structure*.

e) Enter a description.

f) The structure contains fields used by the function module.

g) Add components using the data provided in the table above.

h) Save and activate.

Use your package and transport request. Ignore the warning that an enhancement category is missing.

2. Create a table type.

Use the following data:

Field	Value
Data type	ZAIF_S###_Material_TT
Line Type	ZAIF_S###_MATERIAL

a) Choose *Data type* and, in the field beside it, enter the data provided in the step.

b) Choose *Create*.

c) In the dialog, choose *Table Type*.

d) Enter a description.

e) *Line Type* is preselected. In the field beside it, enter the data provided in the step.

f) Save and activate.

Use your package and transport request.

3. Create the SAP structure.

Use the following data:

Field	Value
Data type	ZAIF_S###_SAP_Material
Materials	ZAIF_S###_MATERIAL_TT (name of the table type created above)

Note:
The typing method is always types.

a) Execute transaction `SE11`.

b) In the field beside *Data type*, enter value provided in the step.

c) Choose *Create*.

d) In the dialog, choose *Structure*.

e) Enter a description.

f) Using the value for materials provided in the step, add a component.

g) Save and activate.

Use your package and transport request. Ignore the Warning that an enhancement category is missing.

Task 3: Create an Interface

Create a new interface MAT_CREATE Version 1 in your Namespace S###. Use the newly created structures as SAP and RAW data structure.

1. Define an interface.

Use the following data:

Field	Value
Interface Name	MAT_CREATE
Interface Version	1
Description	Create Material via File
SAP Data Structure	ZAIF_S###_SAP_MATERIAL (use the name of the above created SAP structure)
Raw Data Structure	ZAIF_S###_RAW_MATERIAL (use the name of the above created raw structure)

a) Navigate to AIF Customizing (transaction `/AIF/CUST`).

b) Create a new interface in *Interface Development* → *Define Interfaces*.

c) Select your namespace and create a new interface with *new entries*.

d) Enter the data provided in the table.

e) Save.

If asked, use your package and transport request.

2. Define interface engines.
   Use the following data:

Note:
The file adapter uses AIF's own XML persistence and runtime. Ensure that you correctly set the application and persistence engines.

   a) Navigate to AIF Customizing (transaction `/AIF/CUST`) under *Interface Development → Define Interfaces → Additional Interface Properties → Specify Interface Engines*.

   b) Select your interface.

   c) Change the Application and Persistence engines to XML.
      For a better view of the details, double click on the interface.

   d) Save.
      If asked, use your package and transport request.

Task 4: Define Mappings

1. Create structure mappings.
   Use the following data:

Field	Value
New entry	LINES
Number of Structure Mapping	10
Destination Structure	MATERIALS (select the table containing the materials in the SAP structure)

   a) Navigate to AIF Customizing (transaction `/AIF/CUST`).

   b) Navigate to *Interface Development → Define Structure Mappings*.

   c) In *Select Source Structure*, create a new entry and select the new entry value provided in the step.

   d) Save.
      This table contains the materials.

      If asked, use your package and transport request.

   e) Select the new entry and choose *Assign Destination Structure*.

   f) Choose *new entries*.

   g) Using the data provided in the table, create a new entry.

   h) Save.

2. Create field mappings.
   Use the following data:

Field	Field Name 1	Field Name 2
CLIENTDATA-BASE_UOM	**UOM**	
HEADDATA-IND_SECTOR	**INDUSTRY**	
HEADDATA-MATERIAL	**%S###_**	**MATERIAL**
HEADDATA-MATL_TYPE	**MATERIALTYPE**	

a) Choose *Define Field Mappings*.

b) Enter the data provided in the table.

c) Save.

Note:
The '%' in a field mapping defines a fix value.

3. Create fix values.

   Use the following data:

Field	Value
CLIENTDATAX-BASE_UOM	**X**
HEADDATA-BASIC_VIEW	**X**

a) Choose *Define Fix Values*.

b) Choose *new entries*.

c) Using the data provided in the table, create a new entry.

d) Save.

Task 5: Define Indirect Mappings

Map the material description. The material descriptions are in the same table as the materials. For each material, you do not want all descriptions but only the correct one. You need an indirect mapping to connect them correctly.

1. Create another field mapping.

   Use the following data:

Field	Value
Field in Destination Structure	MATERIALDESCRIPTION
Sub-Table	LINES (the table containing the materials of the raw structure)
Selection Field	MATERIAL
Operator	EQ Equal
Comparison Field	MATERIAL

a) Navigate to AIF Customizing (transaction `/AIF/CUST`).

b) Navigate to *Interface Development → Define Structure Mappings*.

c) Select your interface.

d) Mark the **`LINES`** Line.

e) Double-click on *Assign Destination Structure*. As you only have one at this point, it displays.

f) Choose *Define Field Mappings*.

g) Choose new entries and create the field mappings provided in the table.

Note:

- To choose the material description, you must scroll to the end on the right half of the screen.
- To choose LINES for the sub-table, you must double-click on **`ZAIF_S###_RAW_MATERIAL`** on the left side of the screen. This makes it selectable on the right side of the screen.
- To map the materials to the corresponding description, a selection and comparison field is required to identify the correct entry in the source structure.

h) Save.

Note:
If requested, use your package and transport request.

2. Create an indirect structure mapping for the sub-table.
   Use the following data:

Field	Value
Number of Structure Mapping	**`20`**
Destination Structure	**`MATERIALDESCRIPTION`**
Indirect Mapping	**`X`**

a) Return to Assign Destination Structure.

b) Create a second entry with the button new entries.

c) Enter the data from the table.

3. Define fix values.
   Use the following data:

Field	Value
Field Name	**LANGU**
Value	**E**

a) Navigate to define fix value and create a new entry.

b) Enter the data provided in the table.

c) Save.

4. Define field mapping.
   Use the following data:

   Map the *MATL_DESC* to the *TEXT* field.

Field	Value
Field in Destination Structure	**MATL_DESC**
Field Name1	**TEXT**

a) Double-click on *Define Field Mappings*.

b) Create a new entry.

c) Enter the data from the table.

d) Save.

Task 6: Define an Action

Using the forward navigation in Assign Actions, create a new action to process the files.

1. Create a new action.
   Use the following data:

Field	Value
Action Number	**10**
Namespace	**S###**
Action	**MATERIAL_CREATE**
Record Type	**MATERIALS**  Note: This table contains the materials in the SAP structure.

a) Navigate to AIF Customizing (transaction `/AIF/CUST`).

b) In *Interface Development → Define Structure Mappings*, select your namespace and interface.

c) In *Assign Actions*, create a new entry.

d) Enter the data provided in the Assign Actions table.

e) Save and confirm that you want to create a new action.

2. Add some data to the action.

Use the following data:

Field	Value
Action description	Create material
Commit Mode	W Commit Work and Wait
Commit Level	F After Each Function
Main Component Type	**ZAIF_S###_MATERIAL** (The name of the structure containing the material in the SAP structure)

a) Double click on the action name.

b) Enter the data provided in the table above.

c) Save.

Note:
If asked, use your package and transport request.

3. Create a function for your action.

Use the following data:

Field	Value
Function Number	**10**
Function Module Name	**ZAIF_S###_AC_MATERIAL_CREATE**
Function group	ZAIF_S###
CURR_LINE	
parameter type	ZAIF_S###_MATERIAL

Code

```
DATA: ls_return TYPE bapiret2.

  CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
    EXPORTING
      headdata            = curr_line-headdata
      clientdata          = curr_line-clientdata
      clientdatax         = curr_line-clientdatax
    IMPORTING
      return              = ls_return
    TABLES
      materialdescription = curr_line-materialdescription.
```

```
APPEND ls_return TO return_tab.
```

a) Choose *Define Functions* and create a new entry.

b) Enter the data from the Function table and choose *Enter*.

c) Confirm that you want to create the function module.

d) Enter the function group value provided in the step, and save.

e) Double-click on the function module name and switch to change mode.

f) Delete the line with the text BREAK-POINT.

g) Switch to the changing parameters tab and change the associated type of parameter *CURR_LINE* to the value provided in the step.

Note:
If you used other names for the structure, change the associated type to the main component type used in the action definition.

h) Return to the *Source Code* tab and insert the code provided in the step.

i) Save and activate the function module.

Note:
If asked, use your package and transport request.

j) Close the function module.

Task 7: Configure a File Adapter

You want to pass the material data to AIF via your created file. To read this file, configure the file adapter in AIF Customizing.

1. Configure the file adapter in AIF Customizing.

   Use the following data:

Field	Value
Config. ID	**MAT_CREATE**
File Type	**Text File**
File Content	**Flat Structure**
Text Type	**Special Character as Separator**
Field Separator	In order to get the correct separator open your txt file in notepad and check for the correct separator.
Header Line Exists	**X**
Structure Name	**ZAIF_S###_RAW_MATERIAL** (the name of the raw structure created above)

Field	Value
Field Name	**LINES**

a) Navigate to AIF Customizing (transaction /AIF/CUST)

b) In *System Configuration → Configure File Adapter*, select your namespace and create a new entry.

c) Enter the data provided in the table.

d) Save.

Task 8: Test Your Development

You want to load files to AIF with your file adapter configuration.

1. Load files to AIF.

   Use the following data:

Field	Value
Config. Namespace	**S###**
Config. ID	**MAT_CREATE**

a) In the SAP Menu, select *File Upload* under *SAP Menu → Cross-Application Components → SAP Application Interface Framework → File Upload*.

> Note:
> You can also use transaction /AIF/LFA_UPLOAD_FILE.

b) Enter the data provided in the table.

c) Select *Upload From Directory – Local PC*.

d) In *Physical Path*, select the file created at the beginning of the exercise.

e) Execute.

> Note:
> It might happen, that the following error displays: **Subobject S### MAT_CREATE is not defined**. in this case, ask your trainer.

A summary of the import displays.

2. Check the result.

a) Navigate to Monitoring and Error Handling (transaction /AIF/ERR).

b) Select your namespace and interface.
You see one message.

Unit 11
Exercise 23

Use Additional Features for Simplified Error Handling

Business Scenario

You want to give the users who monitor the interfaces the possibility of viewing the created materials directly. You prefer to skip the first screen and pass the material number directly to the parameter *ID MAT*.

In case an error occurs, you want to give additional information to your users. You force an error by using a non-existent unit of measure and explain this error message. Also, the message text delivered from the function module does not fit the needs of the user monitoring the interface. You want to give them a different message.

At the end of the exercise, the user who monitors this interface wants to know the location of the field causing the error. To provide this information, you decide to use a customer data link.

Note:
This exercise depends on completing the exercise **Use The File Adapter**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Use Additional Features for Simplified Error Handling

1. Create a custom function to display the material.
   Use the following data:

   Transaction to display a material: `MM03`

   Message text: **`The Material S###_MAT_###_# has been created or extended`**

Field	Value
Transaction	`MM03`
Text	Display Material (S###)
Tooltip	Display Material (S###)
Icon	@A6@

2. Change the custom function.

Use the following data:

Field	Value
Parameter ID	MAT
Fill Method	Message Variable
Value From Message Variable	Select [MsgVar1]
Skip First Screen	Select

3. Customize hints.

   Use the following data:

Field	Value
Tooltip	Wrong unit of measure
Enter a text	The unit of measure transferred in the file does not exist in the system. Correct unit of measure in the data content and restart the message.

4. Customize the message text.

   Use the following data:

   Message text: **`Unit of measure &1 does not exist`**

5. Customize a data link.

Unit 11
Solution 23

Use Additional Features for Simplified Error Handling

Business Scenario

You want to give the users who monitor the interfaces the possibility of viewing the created materials directly. You prefer to skip the first screen and pass the material number directly to the parameter *ID MAT*.

In case an error occurs, you want to give additional information to your users. You force an error by using a non-existent unit of measure and explain this error message. Also, the message text delivered from the function module does not fit the needs of the user monitoring the interface. You want to give them a different message.

At the end of the exercise, the user who monitors this interface wants to know the location of the field causing the error. To provide this information, you decide to use a customer data link.

Note:
This exercise depends on completing the exercise **Use The File Adapter**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Use Additional Features for Simplified Error Handling

1. Create a custom function to display the material.
   Use the following data:

   Transaction to display a material: `MM03`

   Message text: **`The Material S###_MAT_###_# has been created or extended`**

Field	Value
Transaction	`MM03`
Text	Display Material (S###)
Tooltip	Display Material (S###)
Icon	@A6@

a) Navigate to Monitoring and Error Handling (transaction `/AIF/ERR`) and select the interface created in exercise 18.

b) Display all messages of the interface.

c) Select a successful message.

d) In *Log Messages*, select the message text provided in the step.

e) Choose the *Functions* column.

f) In the dialog, enter the data from the table above.

g) Save.

h) Re-check the message.
In *Log Messages*, the new icon for the transaction displays in the Functions column.

i) Choose the new icon.
The display material transaction opens in a new window.

2. Change the custom function.
Use the following data:

Field	Value
Parameter ID	MAT
Fill Method	Message Variable
Value From Message Variable	Select [MsgVar1]
Skip First Screen	Select

a) Return to Monitoring and Error Handling (transaction `/AIF/ERR`) and select the interface created in exercise 18.

b) Navigate to the log message where the custom function was created.

c) In the menu of *Log Messages*, choose *Customize* → *Custom Functions*.

d) In the dialog, select your custom function and choose *Edit*.

e) In *Create Function*, using the data provided in the Parameter table, add a parameter.

f) Choose *Test Function*.
You are forwarded to the `DISPLAY MATERIAL` transaction. The first screen is skipped. The material displays directly.

g) Return and save the changes.

h) To test the function directly from the menu, in *Log Messages*, choose the *Test Function* icon and test your development.
You are forwarded directly to the material.

3. Customize hints.
Use the following data:

Field	Value
Tooltip	Wrong unit of measure

Field	Value
Enter a text	The unit of measure transferred in the file does not exist in the system. Correct unit of measure in the data content and restart the message.

a) To prepare this customization, open the file created in exercise 23 and change the entry in the Unit field to XX for the first entry.

b) Upload the file using transaction /AIF/LFA_UPLOAD_FILE. Check the result in Monitoring and Error Handling. You should see an erroneous message.

c) Go to Monitoring and Error Handling (transaction /AIF/ERR) and select the interface created in exercise 23.

d) Display all messages of the interface.

e) Select the erroneous message created in the step before.

f) In the Log Messages view, select the log message and click on the Hints column.

g) In the dialog, enter the data from the table.

h) Save.

i) Choose the icon in the *Hints* column and test the hint's display.

4. Customize the message text.

   Use the following data:

   Message text: **`Unit of measure &1 does not exist`**

   a) In *Monitoring and Error Handling*, select the erroneous message.

   b) Choose *Customize* → *Custom Message Text*.

   c) Enter a new message using the message text provided in the step.

   d) Save.

   Note:
   You can also pass the content of the message variables to your custom message text. Add the corresponding placeholders, for example, **`&1`**, **`&2`**, **`&3`**, or **`&4`**.

   You see the changed message text in Monitoring and Error Handling.

5. Customize a data link.

   a) In Monitoring and Error Handling, select the erroneous message and the Column you are linking to it. Choose *Customize* → *Custom Data Link*.

   b) In the *Custom Data Link* dialog, choose *Create*.
   This creates a new data link.

   c) Double click on the message.
   The plant field highlights.

Unit 12
Exercise 24

Maintain and Use Interface Variants

Business Scenario

It is required, that you know about the use of interface variants.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Note:
This exercise depends on completing the exercises:

- Perform Interface Monitoring Without AIF
- Manage IDoc Errors
- Create Basic AIF Elements
- Create and Test Mapping for a Sales Agent Structure
- Create Checks and Fix Values
- Map Structure Indirectly For Flight Bookings With Value Conversion
- Create a Value Mapping
- Create A Conditional Mapping
- Create Actions
- Set Additional Actions

Task 1: Create an Interface Variant

You want to change the processing of the FLBOOKING interface for sales agency 120. Sales Agency 120 is restricted to only book flights departing at least three months in the future.

1. Add an additional check for agency 120.

   Use the following data:

Field	Value
Interface Variant	**`FLBOOKING_AGENCY`**
Variant Description	**`FLBOOKING Sales Agency Variant`**

Task 2: Create a Variant Assigning Table

You want to create a variant assignment table with name **ZS###_VAR** to make the decision if the variant is used or not. AIF delivers the template table `/AIF/VAR_AS_TMPL` which you can copy and add the field *SALESAGENCY* with the Data Element *S_AGENCY*.

1. Create a variant assigning table field.
   Use the following data:

Field	Value
Name of the table	ZS###_VAR
SALESAGENCY	of the type *S_AGENCY*

2. Change the maintenance mode.
   This change enables new table entries to be entered direct with transaction `SE16`.
   Use the following data:
   Data Browser / Table View Maint.: **Display / Maintenance Allowed**
3. Create a new entry in the variant assigning table.

> Note:
> You need an entry in this table for every agency that uses the variant.

   Use the following data:

Field	Value
Namespace	S###
Interface Name	FLBOOKING
Interface Version	1
Variant Namespace	S###
Variant Name	FLBOOKING_AGENCY
Salesagency	120

Task 3: Define Variant Assigning Tables

1. Connect the variant table to your interface.
   Use the following data:
   Name of Variant Assigning Table: **ZS###_VAR**

Task 4: Define Variant Key Fields

Specify SALESAGENCY as variant key field for your interface. Assign it to field AGENCYNUM of structure TRAVEL_AGENCY.

1. Define interface key fields.
   Use the following data:

Field	Value
Key Field Number	10
Interface Variant Key Field	SALESAGENCY
Field Name	TRAVEL_AGENCY-AGENCYNUM

Task 5: Define Variant Mappings

1. Create a new FLBOOKING interface entry.
2. Create a new variant.

Field	Value
Source Structure	FLIGHT_BOOKINGS
Number of Structure Mapping	10
Destination Structure	FLIGHT_BOOKINGS
Indirect Mapping	X

3. Create the additional check for the flight data that you want to use in the variant.

Field	Value
Assign Checks	
Number of Check	**10**
Namespace	**S###**
Check	**CHECK_FLDATE**
Check Raw Data	**X**
Ignore Data if Check is not Successful	**Treat as error if check is not successful**
Fieldname 1	**FLDATE**
Check	
Check description	Check if FLDATE is in the future
Error Msg. Class	**ZAIF_TRAINING**
Error Msg. Number	**010**
Variable Definition	**$1**

4. Use the function module ZAIF_BIT750_CHECK_DATE as the single check number 10.

Field	Value
Number of Check	**10**
Check Description	Check if FLDATE is in the future

Field	Value
Check Function Module	**ZAIF_BIT750_CHECK_DATE**

Task 6: Test Your Development

1. Create different messages for your FLBOOKING interface via the AIF test tool.
   Use the following data:

Field	Value
Sales Agency	**120**
Used sales agency	**109**
Flight departure	Next three months

Unit 12
Solution 24

Maintain and Use Interface Variants

Business Scenario

It is required, that you know about the use of interface variants.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Note:
This exercise depends on completing the exercises:

- Perform Interface Monitoring Without AIF
- Manage IDoc Errors
- Create Basic AIF Elements
- Create and Test Mapping for a Sales Agent Structure
- Create Checks and Fix Values
- Map Structure Indirectly For Flight Bookings With Value Conversion
- Create a Value Mapping
- Create A Conditional Mapping
- Create Actions
- Set Additional Actions

Task 1: Create an Interface Variant

You want to change the processing of the FLBOOKING interface for sales agency 120. Sales Agency 120 is restricted to only book flights departing at least three months in the future.

1. Add an additional check for agency 120.

   Use the following data:

Field	Value
Interface Variant	**`FLBOOKING_AGENCY`**
Variant Description	**`FLBOOKING Sales Agency Variant`**

a) Navigate to AIF customizing (transaction `/AIF/CUST`) under *SAP Application Interface Framework → Interface Development → Interface Variants → Define Interface Variants*.

b) Select your namespace and create a new entry.

c) Enter the data provided in the table.

d) Save.

Note:
If asked, use your package and transport request.

Task 2: Create a Variant Assigning Table

You want to create a variant assignment table with name **ZS###_VAR** to make the decision if the variant is used or not. AIF delivers the template table `/AIF/VAR_AS_TMPL` which you can copy and add the field *SALESAGENCY* with the Data Element *S_AGENCY*.

1. Create a variant assigning table field.

   Use the following data:

Field	Value
Name of the table	ZS###_VAR
SALESAGENCY	of the type *S_AGENCY*

   a) Open `SE11`.

   b) Select database table and enter `/AIF/VAR_AS_TMPL` in the field beside it.

   c) Choose the copy icon.

   d) Copy the basic version of the variant assigning table `/AIF/VAR_AS_TMPL`.

   e) In the dialog, enter the name of the table provided in the table above.

   f) Continue.

   g) Choose *Change* to open the new table.

   h) Insert the value of the *SALESAGENCY* provided in the step.

2. Change the maintenance mode.

   This change enables new table entries to be entered direct with transaction `SE16`.

   Use the following data:

   Data Browser / Table View Maint.: **`Display / Maintenance Allowed`**

   a) In *Delivery and Maintenance*, change the value for *Data Browser / Table View Maint.* to the value provided in the step.

   b) Save and activate the table.

3. Create a new entry in the variant assigning table.

Note:
You need an entry in this table for every agency that uses the variant.

Use the following data:

Field	Value
Namespace	S###
Interface Name	FLBOOKING
Interface Version	1
Variant Namespace	S###
Variant Name	FLBOOKING_AGENCY
Salesagency	120

a) Open `SE16` and create a new entry.

b) Enter the data provided in the step.

c) Save.

Task 3: Define Variant Assigning Tables

1. Connect the variant table to your interface.
   Use the following data:
   Name of Variant Assigning Table: **`ZS###_VAR`**

   a) Navigate to AIF customizing (transaction /AIF/CUST) and choose *SAP Application Interface Framework → Interface Development → Interface Variants → Define Assigning Tables*.

   b) Select your namespace.

   c) Select your *FLBOOKING* interface.

   d) Enter the name of the variant assigning table provided in the step for *Name of Variant Assigning Table*.

   e) Save.

Task 4: Define Variant Key Fields

Specify SALESAGENCY as variant key field for your interface. Assign it to field AGENCYNUM of structure TRAVEL_AGENCY.

1. Define interface key fields.
   Use the following data:

Field	Value
Key Field Number	10
Interface Variant Key Field	SALESAGENCY

Field	Value
Field Name	TRAVEL_AGENCY-AGENCYNUM

a) Open AIF customizing (transaction `/AIF/CUST`).

b) Navigate to *SAP Application Interface Framework → Interface Development → Interface Variants → Define Interface Key Fields*.

c) Select your namespace, interface name and version.

d) Using the data from the table, create a new entry.

e) Save.

Task 5: Define Variant Mappings

1. Create a new FLBOOKING interface entry.
   a) Navigate to AIF customizing (transaction `/AIF/CUST`).

   b) Choose *SAP Application Interface Framework → Interface Development → Interface Variants → Define Variant Mappings*.

   c) Select your namespace and interface variant.

   d) Create a new entry for your FLBOOKING interface.

2. Create a new variant.

Field	Value
Source Structure	FLIGHT_BOOKINGS
Number of Structure Mapping	10
Destination Structure	FLIGHT_BOOKINGS
Indirect Mapping	X

   a) Enter namespace, interface name and version and save the entry.

   b) Select *Add/Select Structure Mapping*.

   Use the same structure mapping you already made for the FLIGHT_BOOKINGS structure. However, you must create it under the variant again.

   c) Using the data from the table, create a new entry.

   d) Save.

3. Create the additional check for the flight data that you want to use in the variant.

Field	Value
Assign Checks	
Number of Check	**10**
Namespace	**S###**
Check	**CHECK_FLDATE**

Field	Value
Check Raw Data	**X**
Ignore Data if Check is not Successful	**Treat as error if check is not successful**
Fieldname 1	**FLDATE**
Check	
Check description	Check if FLDATE is in the future
Error Msg. Class	**ZAIF_TRAINING**
Error Msg. Number	**010**
Variable Definition	**$1**

a) Switch to *Assign Checks*.

b) Using the data from the Assign Checks table, create a new entry.

c) Create the check by double-clicking on the name of the check.

d) Confirm that you want to create the check.

e) Select the new check.

f) Define the data from the Check table.

g) Save.

4. Use the function module ZAIF_BIT750_CHECK_DATE as the single check number 10.

Field	Value
Number of Check	**10**
Check Description	Check if FLDATE is in the future
Check Function Module	**ZAIF_BIT750_CHECK_DATE**

a) Navigate to Define Single Checks.

b) Create a new entry and enter the data from the table above.

c) Save.

Task 6: Test Your Development

1. Create different messages for your FLBOOKING interface via the AIF test tool.
   Use the following data:

Field	Value
Sales Agency	**120**
Used sales agency	**109**
Flight departure	Next three months

a) Navigate to AIF test tool (transaction `/AIF/IFTEST`) and select your test file.

b) Change the Sales Agency to the value created in the step.

c) In the *FLIGHT_BOOKINGS* structure, enter at least one flight that will depart within the next 3 months.
This receives the error created in the check.

d) Process the file via the XML runtime and check the result in the monitoring and error handling transaction (transaction `/AIF/ERR`).
You see the error specified in the *CHECK_DATE* check.

e) Using the data provided in the table, send a second message via the AIF test tool.

f) Process the file in the XML runtime and check the result in the monitoring and error handling.
You do not receive the error specified in the *CHECK_DATE* check (the date restriction is only valid for sales agency 120).

Unit 13
Exercise 25

Use the Analyzer

Business Scenario

You created a test file in the AIF Test Transaction for one of the interfaces that work with the AIF XML Engines.

You want to discover what happened to your interface. To do this, you use the analyzer functionality from the test tool.

Note:
This exercise depends on completing the exercises

- Perform Interface Monitoring Without AIF
- Manage IDoc Errors
- Create Basic AIF Elements
- Create and Test Mapping for a Sales Agent Structure
- Create Checks and Fix Values
- Map Structure Indirectly For Flight Bookings With Value Conversion
- Create a Value Mapping
- Create A Conditional Mapping
- Create Actions
- Set Additional Actions

Alternatively complete the exercise **Use The File Adapter**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Use the Analyzer

1. Investigate what happened to your interface.
2. Explore the various functions of the analyzer.

Unit 13
Solution 25

Use the Analyzer

Business Scenario

You created a test file in the AIF Test Transaction for one of the interfaces that work with the AIF XML Engines.

You want to discover what happened to your interface. To do this, you use the analyzer functionality from the test tool.

Note:
This exercise depends on completing the exercises

- Perform Interface Monitoring Without AIF
- Manage IDoc Errors
- Create Basic AIF Elements
- Create and Test Mapping for a Sales Agent Structure
- Create Checks and Fix Values
- Map Structure Indirectly For Flight Bookings With Value Conversion
- Create a Value Mapping
- Create A Conditional Mapping
- Create Actions
- Set Additional Actions

Alternatively complete the exercise **Use The File Adapter**.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Use the Analyzer

1. Investigate what happened to your interface.
   a) Open transaction `/AIF/IFTEST` and search your test file.
   b) Double-click on your file and choose *Read Data*.
   c) On the next screen, choose *Analyze* on the menu bar.

You are forwarded to the AIF Analyzer.

d) Double click on the interface name in the process step window in the middle of the screen.

Note:
Use the *Next Step* and *Previous Step* buttons to navigate through the different processing steps of the interface.

e) In the RAW structure window, select the *FLDATE* field of the *FLIGHT_BOOKINGS* structure.

f) Right click on the *FLDATE* field and select **`'Show DDIC'`**.
You are forwarded to the DDIC element.

g) Return to the Analyzer.

h) Choose *FLDATE* in the RAW structure again. Use *Show where field is used* in the menu bar.
Processing steps and fields where the value of the FLDATE field of the RAW structure is used are highlighted.

2. Explore the various functions of the analyzer.

a) Use the various buttons.

Unit 14
Exercise 26

Maintain an Interface Determination with the Example of an Outbound IDoc

Business Scenario

You want to process an outbound IDoc via AIF. You require an ABAP PSS port (type ABAP_PI) that uses `/AIF/SINGLE_IDOC_PORT_FUNCTION` as the function module. Check if the port exists.

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc. This is done with the AIF IDoc generation report (transaction `/AIF/IDOC_GEN`). The report also creates your interface.

During the processing of the IDoc, AIF is called and index table and message index table entries are written. It is possible to display data in the Interface Monitor. To display the messages in the Interface Monitor, maintain a recipient. You can reuse the already existing ALL_INTERFACES recipient.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create an IDoc Structure

1. Check the port function.
2. Generate a structure and interface.
   Use the following data:

Field	Value
General Details area	
Basic Type	**MATMAS05**
Message Type	**MATMAS**
Extension	<remains empty>
Structure Prefix	**ZAIF_S###_**
Root Structure Name	**ZAIF_S###_MATMAS_OUTBOUND**
Create or Update Structure	<flag>
Create Interface	<flag>, when flagged, hit *Enter* to make the other fields visible
Structure Details area	

Field	Value
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**
Interface Name	**OUT1_S###**
Interface Version	**1**
Interface Description	**Outbound Interface 1**
Interface Direction	**both**
IDoc Processing Scenario	**6 (AIF Runtime; Call IDoc function in action; Pre-processing)**

3. Check the engines used for the interface.
   Use the following data:

Field	Value
Interface Name	OUT1_S###
Interface Engines	
Application and Persistence Engine	**IDoc**
Selection Engine	**AIF Index Tables**
Logging Engine	**AIF Application Log**
IDoc Types	
Message Type	MATMAS
Basic Type	MATMAS05

4. Map Header Data to the IDoc being sent from AIF.
   You can use the fix Value ### for the Field *EDIDC-MESCOD*.
   Use the following data:

Field	Value
Number of Structure Mapping	**10**
Destination Structure	**EDIDC**
Fix Value	MESCOD: ### (your namespace without the leading S)
Move Corresponding Fields	<flag>

5. Maintain partner profile.

Note:
You need two entries. One for this IDoc that is sent to AIF and one for the IDoc that is sent to the real partner.

Use the following data:

Field	Value
Logical system	Partner Type LS - ZAIF_S###
First Outbound Options	
Receiver Port	**ZAIF_PORT**
Basic Type	**MATMAS05**
Message Type	**MATMAS**
Pass Immediately	<flag>
Second Outbound Options	
Receiver Port	**ZAIF_FILE**
Message Type	**MATMAS**
Basic Type	**MATMAS05**
Message Code	###
Pass Immediately	<flag>

6. Create an action that triggers the second IDoc.
   Use function module `/AIF/CALL_MASTER_IDOC_DIST`.

   Use the following data:

Field	Value
Action Number	**10**
Namespace	**S###**
Action	**AC_MATERIAL_OUT**
Action	
Action Description	**Send Material to receiver**
Commit Mode	**Commit Work**
Commit Level	**After each function**
Main Component Type	Empty (this means take root)
Define Functions	
Function Number	**10**
Function Module Name	**ZAIF_S###_SEND_IDOC**

Code

```
CALL FUNCTION '/AIF/CALL_MASTER_IDOC_DIST'
  EXPORTING
    TESTRUN              = SPACE
*   SENDING_SYSTEM       =
  TABLES
    RETURN_TAB           = return_tab
  CHANGING
    DATA                 = Data
    CURR_LINE            = Curr_line
    SUCCESS              = Success
    OLD_MESSAGES         = Old_messages.
```

7. Create an interface determination for your IDoc.

   As we all use the MATMAS/MATMAS05 IDoc structure for this exercise AIF does not know, which interface should be taken. To tell AIF each interface needs an interface determination.

   We need a field in the IDoc that is always different. In our case we choose the Logical System ZAIF_S### for this.

   Use the following data:

Field	Value
Basis Type	MATMAS05
Message Type	MATMAS
Field-category	IDoc Control Record
Field-Name	RCVPRN

8. Assign Interfaces

   Use the following data:

Field	Value
Value number	10
Operator	EQUAL
Value for Interface Determination	ZAIF_S###
Namespace	S###
Interface Name	OUT1_S###
Interface Version	1

9. Optional: maintain a recipient.

   If your user is still assigned to the ALL_INTERFACES recipient, you can skip this step.

Task 2: Test Your Development

1. Test your development.

   Transaction for test IDoc: `BD10`.

   Use the following data:

Field	Value
Material	**92**
Message Type	**MATMAS**
Logical system	**ZAIF_S###**
Send material in full	empty

2. Check the created IDocs.
   Check the result in the following transactions:
   - AIF Monitoring and Error Handling transaction (transaction `/AIF/ERR`)
   - `BD87`

Unit 14
Solution 26

Maintain an Interface Determination with the Example of an Outbound IDoc

Business Scenario

You want to process an outbound IDoc via AIF. You require an ABAP PSS port (type ABAP_PI) that uses `/AIF/SINGLE_IDOC_PORT_FUNCTION` as the function module. Check if the port exists.

Before an IDoc can be monitored via AIF, you must create a DDIC structure out of the basic type of the IDoc. This is done with the AIF IDoc generation report (transaction `/AIF/IDOC_GEN`). The report also creates your interface.

During the processing of the IDoc, AIF is called and index table and message index table entries are written. It is possible to display data in the Interface Monitor. To display the messages in the Interface Monitor, maintain a recipient. You can reuse the already existing ALL_INTERFACES recipient.

Note:
In this exercise, when a value or an object name includes ###, replace ### with the number that your instructor assigned to you.

Task 1: Create an IDoc Structure

1. Check the port function.
   a) Navigate to transaction `WE21`.
   b) Expand the *ABAP-PI* node.
   c) Double-click on *ZAIF_PORT*.
   It uses /AIF/SINGLE_IDOC_PORT_FUNCTION as function module.
2. Generate a structure and interface.
   Use the following data:

Field	Value
General Details area	
Basic Type	**MATMAS05**
Message Type	**MATMAS**
Extension	<remains empty>
Structure Prefix	**ZAIF_S###_**
Root Structure Name	**ZAIF_S###_MATMAS_OUTBOUND**

Field	Value
Create or Update Structure	<flag>
Create Interface	<flag>, when flagged, hit *Enter* to make the other fields visible
Structure Details area	
Structure Prefix	ZAIF_S###_
Package	ZAIF_S###
Workbench Request	<use the request given to you from the instructor>
Interface Definition area	
Namespace	**S###**
Interface Name	**OUT1_S###**
Interface Version	**1**
Interface Description	**Outbound Interface 1**
Interface Direction	**both**
IDoc Processing Scenario	**6 (AIF Runtime; Call IDoc function in action; Pre-processing)**

a) Navigate to the AIF IDoc generation report (transaction `/AIF/IDOC_GEN`).

b) Generate an interface for the MATMAS IDoc.

c) Enter the above data in the report.

Hint:
Some of them are only shown after selecting the *Create Interface* and the *Create or Update Structures* checkbox.

d) Add the workbench and the customizing request given to you by the trainer.

3. Check the engines used for the interface.

Use the following data:

Field	Value
Interface Name	OUT1_S###
Interface Engines	
Application and Persistence Engine	**IDoc**
Selection Engine	**AIF Index Tables**
Logging Engine	**AIF Application Log**
IDoc Types	
Message Type	MATMAS

Field	Value
Basic Type	MATMAS05

a) Navigate to AIF customizing (transaction `/AIF/CUST`) under *Define Interfaces* and choose your namespace.

b) Double click on the interface name provided in the table above.
It uses the same Structure *ZAIF_S###_MATMAS_OUTBOUND* as SAP and as RAW Structure.

c) Ensure the *Move Corresponding Flag* is set.

d) Navigate to AIF customizing (transaction `/AIF/CUST`) under *Additional Interface Properties → Specify Interface Engines* and select your namespace.

e) Ensure the settings have the value provided in the Interface Engines table.

f) Using the values provided in the IDoc Types, check the assigned IDoc types.

g) Save.

4. Map Header Data to the IDoc being sent from AIF.
You can use the fix Value ### for the Field *EDIDC-MESCOD*.
Use the following data:

Field	Value
Number of Structure Mapping	**10**
Destination Structure	**EDIDC**
Fix Value	MESCOD: ### (your namespace without the leading S)
Move Corresponding Fields	<flag>

a) Navigate to AIF customizing (transaction `/AIF/CUST`) under *Interface Development → Define Structure Mappings* and select your outbound IDoc Interface.

b) Create a new entry and select the EDIDC structure.

c) Select the new entry, navigate to *Assign Destination structure*, and create a new entry.

d) Enter the data provided in the table.

e) In *Fix values*, using the value for *MESCOD* provided in the step, create the entry.

f) Save.

5. Maintain partner profile.

Note:
You need two entries. One for this IDoc that is sent to AIF and one for the IDoc that is sent to the real partner.

Use the following data:

Field	Value
Logical system	Partner Type LS - ZAIF_S###
First Outbound Options	
Receiver Port	**ZAIF_PORT**
Basic Type	**MATMAS05**
Message Type	**MATMAS**
Pass Immediately	<flag>
Second Outbound Options	
Receiver Port	**ZAIF_FILE**
Message Type	**MATMAS**
Basic Type	**MATMAS05**
Message Code	*###*
Pass Immediately	<flag>

a) Execute transaction `WE20` and open the node *Logical Systems*.

b) Select your logical system provided in the step.

c) Select create outbound parameters.

d) In outbound options, enter the data provided in the First Outbound Options table.

e) Save.

f) Select *Create Outbound Parameters*.

g) Enter the data provided in the table above.

h) In *Outbound Options*, enter the data provided in the table, below the **Second Outbound Options** row.

i) Save.

6. Create an action that triggers the second IDoc.
   Use function module `/AIF/CALL_MASTER_IDOC_DIST`.

   Use the following data:

Field	Value
Action Number	**10**
Namespace	**S###**
Action	**AC_MATERIAL_OUT**
Action	
Action Description	**Send Material to receiver**
Commit Mode	**Commit Work**

Field	Value
Commit Level	**`After each function`**
Main Component Type	Empty (this means take root)
Define Functions	
Function Number	**`10`**
Function Module Name	**`ZAIF_S###_SEND_IDOC`**

Code

```
CALL FUNCTION '/AIF/CALL_MASTER_IDOC_DIST'
  EXPORTING
    TESTRUN              = SPACE
*   SENDING_SYSTEM       =
  TABLES
    RETURN_TAB           = return_tab
  CHANGING
    DATA                 = Data
    CURR_LINE            = Curr_line
    SUCCESS              = Success
    OLD_MESSAGES         = Old_messages.
```

a) Navigate to AIF customizing (transaction `/AIF/CUST`) under *Interface Development → Define Structure Mappings*. Select your first outbound interface and navigate to *Assign Actions*.

b) Create a new entry and enter the information from the Assign Actions table.

c) Confirm that you want to create the action and save.

d) Double click on action name. Switch to change mode and navigate to your new action. Enter the information from the Action table.

e) In *Define Functions*, using the data from the Define Functions table, create a new entry.

f) Save.

g) In *Define Functions*, using the data from the Define Functions table, create a new entry.

h) Press Enter. A dialog appears asking would you like to create the function module.

i) Choose *Yes*.

j) Insert your function group **`ZAIF_S###`** and save.

k) Double click on the function module. Transaction SE37 starts in a new window.
In this function module, the AIF Function for sending an IDoc out of AIF (`/AIF/CALL_MASTER_IDOC_DIST`) must be called.

l) Add the code provided in the step.

7. Create an interface determination for your IDoc.
As we all use the MATMAS/MATMAS05 IDoc structure for this exercise AIF does not know, which interface should be taken. To tell AIF each interface needs an interface determination.

We need a field in the IDoc that is always different. In our case we choose the Logical System ZAIF_S### for this.

Use the following data:

Field	Value
Basis Type	MATMAS05
Message Type	MATMAS
Field-category	IDoc Control Record
Field-Name	RCVPRN

a) Execute transaction `/AIF/CUST`.

b) Choose the menu *System-configuration → Interface Determination → Define Interface Determination für IDoc Interfaces → Define Determination Key*.

c) Choose *New Entries*.

d) Fill in your *Basis Type* **`MATMAS05`** and your *Message Type* **`MATMAS`**.

e) In the *Field-category* choose **`IDoc Control Record`** using F4.

f) For the *Field-Name* choose **`RCVPRN`** using F4.

g) Save.

8. Assign Interfaces

Use the following data:

Field	Value
Value number	10
Operator	EQUAL
Value for Interface Determination	ZAIF_S###
Namespace	S###
Interface Name	OUT1_S###
Interface Version	1

a) On the left hand menu choose *Assign Interfaces*.

b) Choose *New Entries*.

c) Save.

9. Optional: maintain a recipient.

If your user is still assigned to the ALL_INTERFACES recipient, you can skip this step.

a) Navigate to *System Configuration → Assign Recipients*

b) Assign your user to the recipient.

Task 2: Test Your Development

1. Test your development.

   Transcription for test IDoc: `BD10`.

   Use the following data:

Field	Value
Material	**92**
Message Type	**MATMAS**
Logical system	**ZAIF_S###**
Send material in full	empty

   a) Execute transaction `BD10`.

   b) Enter the data from the table.

   c) Execute.

   Two dialogs appear telling you what you create. Click them away.

2. Check the created IDocs.

   Check the result in the following transactions:

   - AIF Monitoring and Error Handling transaction (transaction `/AIF/ERR`)
   - `BD87`

   a) Start the Interface Monitor via transaction `/AIF/IFMON`.

   b) On the right side, select and expand your namespace.
   Your *OUT1_S###* interface displays as a sub node.

   c) To see the interfaces, double-click *Sum*.
   You see an interface for each IDoc you sent with BD10.

   d) Navigate to transaction `BD87`.

   e) Choose *Execute*.
   You find all IDocs sent today.

   f) Check for your IDocs.
   You now have two IDocs:

   - An IDoc sent from BD10 to AIF
   - An IDoc sent from AIF during the action to the “real” receiver

   To divide them, look at the header data. The second one contains the message function ###.