IDoc Interface / Electronic Data Interchange (BC-SRV-EDI)

HELP.BCSRVEDI

Release 4.6C

Copyright

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Contents

IDoc Interface / Electronic Data Interchange (BC-SRV-EDI)

IDoc Interface / Electronic Data Interchange (BC-SRV-EDI)

Purpose

Business data is exchanged with an external system using the IDoc Interface.

The IDoc Interface consists of the definition of a data structure and a processing logic for this.

The data structure is the IDoc. It is the exchange format that unites the communicating systems. Using IDocs you can define an exception handling within the R/3 System using SAP Business Workflow, without the need for the data to already be an SAP application document.

You require the IDoc Interface in the following scenarios:

- Electronic Data Exchange (EDI)
- Application Link Enabling (ALE)
- Linking to any other business application systems (for example, PC applications, external workflow tools) using IDoc.

Features

You can access all the functions cited from the **Initial node of the IDoc Interface**: From the R/3 initial screen choose *Tools* → *Business Communication* → *IDoc* → *IDoc Basis* ( ).

Processing IDocs [Page 10]

Describes the various inbound and outbound processing paths and status processing. This section is intended for both administrators and end users.

Configuring Ports [Page 37] .

Describes the technical link to the external system, right down to the operating system level. Ports must be configured before data exchange with the external system can take place. This section is intended for administrators.

Defining a Partner [Page 68]

The partner profiles are a further prerequisite for data exchange. This involves defining who can exchange messages with the R/3 System and via which port. This section is intended for administrators.

Processing Tests [Page 91]

The IDoc Interface provides tools for testing IDoc processing. These tests should be performed when new messages are used and when new IDoc types are defined. This section is intended for administrators.

Monitoring [Page 105]

Both passive (display processing) and active (dispatch warnings and notes) monitoring functions are documented here. This section is intended for both administrators and application end users.

Archiving IDocs [Page 124]

The options available for archiving IDocs are described here. This section is intended for administrators.

IDoc Interface / Electronic Data Interchange (BC-SRV-EDI)

Structure, Documentation and Definition of IDoc Types [Page 141]

Customer extensions to IDoc types are described here. This section is intended for R/3 developers and administrators.

General configuration

IDoc Administration: User Parameters [Page 247]

This section describes those IDoc administration parameters which are changed frequently for configuration purposes while the application is in use. This section is, of course, intended for administrators.

Additional Settings [Page 248]

There are other ways of configuring the working environment for your IDoc Interface, although this is not usually necessary. These options are listed here. This section is intended for administrators.

Processing IDocs

Use

The business data is saved in IDoc format in the IDoc interface and is forwarded as IDocs. If an error occurs, exception handling is triggered via workflow tasks. The agents who are responsible for these tasks and have the relevant authorizations are defined in the IDoc interface.

Features

The IDoc interface supports three types of data flow with the external system:

- Outbound processing [Page 11]

  IDocs are transferred to a receiving system from your SAP System.

- Inbound processing [Page 18]

  IDocs are transferred to your SAP System from an upstream system.

- Status processing [Page 23]

  The receiving system confirms the processing status of outbound IDocs to your SAP System.

Control records and data records are sent for IDoc inbound processing and IDoc outbound processing. Status records are sent in the status confirmation data flow (exception: status confirmation via the specific IDoc type SYSTAT01).

See also:

Exception Handling [Page 26]

Exception handling functions are implemented when errors occur.

Role Resolution in Exception Handling [Page 30]

This section describes how the agents responsible for a work item are determined in the IDoc interface.

Communication with Older Releases [Page 33]

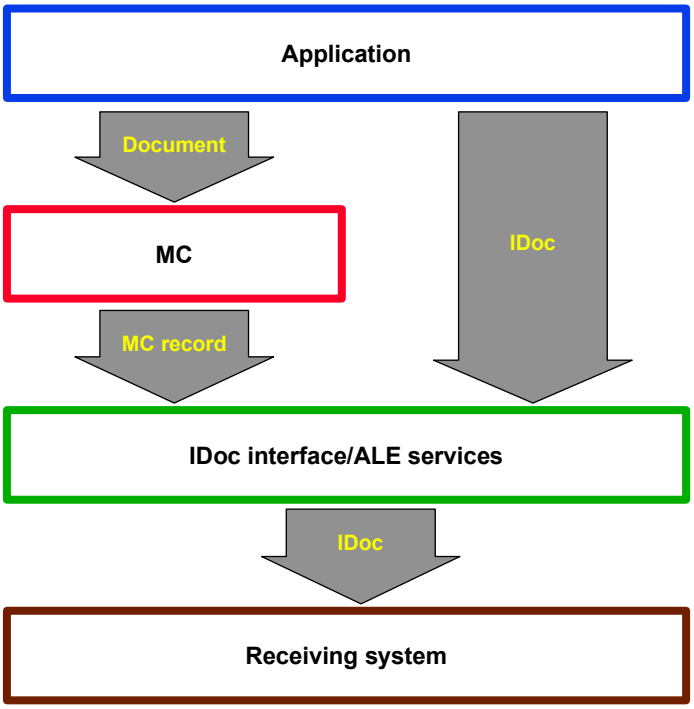
Additional customizing settings may be required.

Outbound Processing

Use

In outbound processing, document data is written to IDocs and sent to the receiving system.

Features



The message can be sent from the application to the IDoc interface along two different paths:

- The indirect path using Message Control [Page 12] (MC): a series of conditions are checked to “find” the message. If one of these conditions is fulfilled, the message that was “found” is forwarded to the IDoc interface via the corresponding Message Control record. This condition technique also allows the Message Control module to find and send more than one message. For further information on MC see R/3 library → Cross-application functions → Message Control [Ext.].
- the direct path from the application to the interface [Page 15]. Here, the application generates an IDoc which is transferred to the IDoc interface via the function module MASTER_IDOC_DISTRIBUTE.

The IDoc interface can send the IDocs to the receiving system along different paths (receiver ports). Port selection also depends on the receiving system and the hardware used for the installation. Please read the information in the following section

Communication with Older Releases [Page 33]

Outbound Processing under Message Control (MC)

Use

Messages, for example purchase orders, can be found and processed via the Message Control module (MC) in SD and MM. In the case of IDoc processing, that means that the application data is written to IDocs.

Prerequisites

Setting up IDoc processing always requires you to define your partner [Page 68]. In particular for MC, you must assign the application and the MC output type uniquely to an IDoc type in partner profiles. You do this with the additional outbound parameters under MC [Page 79].

Activities

- The Message Control module “finds” one or more messages by selecting those that match the relevant business process from a set of predefined messages. The messages are defined in the application in so-called condition tables. The messages found are “proposed” by the MC: That can be several per document. In many applications you can view and modify (“process“) messages before you release the data, post the document and send the messages as IDocs.

  In the specified case (message processing through IDoc dispatch) the system additionally checks messages found to determine whether the message partner was processed as a partner in the IDoc interface. The message is only proposed if this is the case, and can then be further processed. Many applications provide **determination analysis**, which helps you to trace message determination and locate possible errors.

- The Message Control module can process the messages immediately (after the application document has been updated). You can also process the messages found manually or in the background at a predefined time. Since you can also define the time at which IDocs are to be generated by the IDoc interface, you should combine these two times. These combinations are described in the following section: Procedure [Page 13] .

  An order for the vendor VEND is to be created in Purchasing. This order is to be sent via an EDI subsystem after being written as an IDoc of type ORDERS01. In order to do this, define the output type “NEU” (new) for VEND in purchasing master data and assign the Message Control dispatch time “4” (immediately with application update) and the transmission medium “EDI”. Choose the output mode “Transfer IDoc immediately” and “Start subsystem immediately” for VEND in the partner profiles of the IDoc interface and assign the logical message ORDERS to the combination “Application: Purchasing orders”, “Output type: NEU” (new). IDoc type ORDERS01 is assigned to this logical message.

See also:

Technical Implementation [Page 14] .

Outbound Processing under Message Control: Procedure

1. *Message determination*: Call the **master data** from the application and create the message as a **message- or condition record**, that is to say, you define the conditions under which the message is found and proposed, as well as the **message properties**. For example, you can enter the purchasing organization and the business partner as the conditions and the output medium (in this case 6 for EDI), dispatch time and language in which the message is to be sent as the output properties.
2. *Message processing through IDoc dispatch*: The messages are sent by the Message Control module as defined in the condition record, especially with regard to the selected **dispatch time**. You must also define a dispatch time ("**output mode**") in the partner profiles of the IDoc interface: standard combinations of the two times are shown in the table below. The Message Control parameters from the partner profiles must also match the corresponding fields in the output type. These parameters include:
   - Application
   - Partner
   - Partner function
   - Output type

Dispatch time combinations for the Message Control module and IDoc interface and the EDI equivalents

MC: Dispatch time	IDoc interface: Output mode	EDI equivalent
4 (= immediately)	Send IDoc immediately Start subsystem	Real time
1 (= send with next selection run)	Send IDoc immediately Start subsystem	Fast batch
1	Collect IDocs Start subsystem	Batch
1	Collect IDocs Do not start subsystem	Batch

If you specify that the subsystem (= follow-on system) is not to be started in the partner profiles, the receiving system determines the dispatch time in accordance with the time plan set in the system, that is to say you do not have any control over when the IDoc arrives at the target system.

Outbound Processing Using MC: Technical Background

For a detailed description of the Message Control module, please refer to the documentation under CA → Message Control [Ext.] .

- *Message determination:* The conditions, under which a message is to be found, are stored in the condition tables [Ext.] . These tables are read in an access sequence [Ext.] . The condition tables also contain the **key fields** for the application, that is to say, the fields which the application uses to access the condition records (for example the “purchasing organization” and “vendor” application fields in Purchasing). The condition tables are assigned to an **output type** (for example “NEU” (new) for a purchase order from Purchasing). The output types are combined in Procedures [Ext.] , which are assigned to the application (key, for example “EF” for “purchase order”).

  This organizational structure allows message determination to run in a structured manner and under complex conditions. The output types and tables and the access sequences and procedures are already defined in Customizing for the relevant application.

  The output type is sometimes also referred to as the condition type.

- *Message processing through IDoc dispatch:* The central selection program of the Message Control module, RSNAST00, locates and triggers the form routine EDI_PROCESSING in the program RSNASTED in table TNAPR for the selected output type. EDI_PROCESSING reads the partner profiles and uses the process code to determine the function module which is to generate the IDoc. The process code also determines the type of further processing, for example whether the IDocs are to be processed by the ALE service.

  The function modules for generating the IDocs are usually called IDOC_OUTPUT_<MT>, where <MT> represents the relevant message type. Depending on the **output mode**, the generated IDocs are either collected or forwarded for immediate dispatch. If the IDocs are collected, the report RSEOUT00 must be scheduled in order to forward the IDocs for dispatch.

Direct Outbound Processing

Use

This outbound processing path must be selected for R/3 components which are not linked to the Message Control module. Otherwise it is usually used in ALE scenarios, in which, for example, master data is to be distributed among different R/3 Systems.

Activities

The application is directly responsible for generating the IDoc, that is to say the application data is entered in a certain IDoc type. This can, for example, take place via a separate function module. The recipients are determined by the application or the ALE services. Automatic Message Control is not used here.

In Controlling (CO), a product costing (application component CO-PC) is to be transferred to profitability analysis (CO-PA). The two components are configured in different R/3 Systems, so we have a typical ALE scenario. In the transfer transaction, you select the fields from which the application data is to be written directly to the IDoc, for example material, plant, date and costing variant.

See also:

Direct Outbound Processing: Procedure [Page 16]

Direct Outbound Processing: Implementation for ALE [Page 17]

Direct Outbound Processing: Procedure

Choose the relevant send transaction in the application and enter the parameters accordingly. Make sure that the specified communication parameters (such as the target system) are also maintained as a port in the partner profiles of the IDoc interface. In ALE scenarios a “tRFC” port type should be entered and the partner should be an “LS” type (for “logical system”).

Direct Outbound Processing: Technical Implementation for ALE

The way in which the IDoc is generated depends on the respective application. The following is an example of an ALE scenario in which a function module is responsible for generating the IDocs.

The function module is called in the application transaction. The function module generates a so-called **master IDoc**, which is transferred to the administration module MASTER_IDOC_DISTRIBUTE, which checks the control record and then calls the function module COMMUNICATION_IDOC_CREATE. This module “filters” the master IDoc (that is to say removes any data which is not required for communication). This “filtered” IDoc is referred to as the **communication IDoc** and is forwarded for further processing to the function module EDI_OUTPUT_NEW by MASTER_IDOC_DISTRIBUTE.

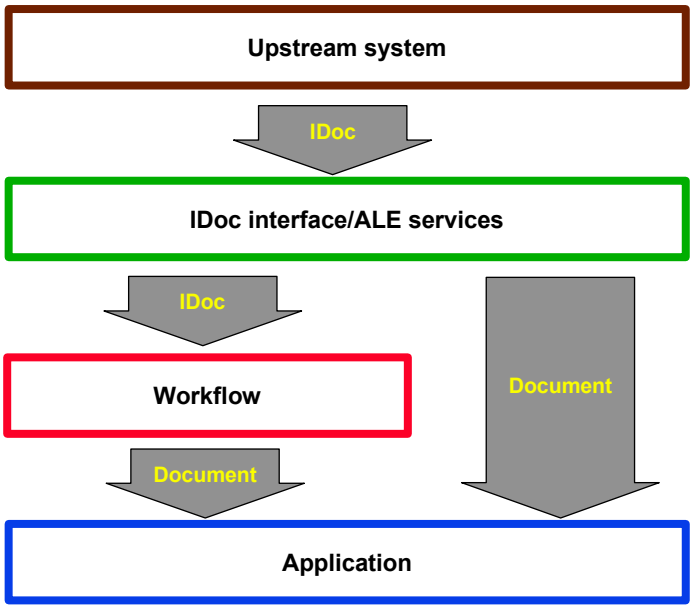
Inbound Processing

Use

In inbound processing, IDocs are transferred to the interface and stored in the R/3 System. The document data is generated in a second step, also in the course of a workflow.

Features

The upstream system transfers an IDoc to the IDoc interface via the R/3 System port. For this reason, you do not have to specify a port in the inbound partner profiles; the IDoc interface only has to “recognize” the upstream system as a port. A **port definition**, which provides a unique ID for the upstream system, must be available for the port. The technical parameters of this port definition can (and usually are) overwritten by the upstream system.



The IDoc is “accepted”, that is, saved in the database, if the upstream system is recognized. If your partner is defined with the corresponding message in your **partner profiles**, the IDoc is then processed further. This is done **independently** in a second step. This ensures that the external system can receive the data quickly and reliably (automatically).

The following paths are available for further processing:

- The direct path via a function module which transfers the IDoc data to the corresponding application document.
- The indirect path via SAP Business Workflow (single- or multistep task). When an IDoc is received, a work item is created as an instance of the corresponding task. The work item appears in the integrated inbox of the selected agent. For further information on SAP Business Workflow see Basis → Business Management → SAP Business Workflow [Ext.].

IDoc type TXTRAW02 is processed via the single-step task TS30000008: A mail is sent to the SAPoffice user (or the organizational unit) that is entered as the recipient in segment E1TXTAD. If this segment is missing, the *permitted agent* is determined from the partner profiles as the recipient. The mail contains the text from the IDoc data records. You can also send mail attributes such as priority or “executability”.

The indirect path in Release 2.1/2.2 is implemented via process technology. This technology is no longer supported.

Activities

Inbound processing: Procedure [Page 20]

Inbound processing: Implementation [Page 21] .

Inbound Processing: Procedure

Purpose

Therefore, you must always configure inbound processing when you want to implement new business processes where data will be received by IDoc. An example is EDI inbound processing of standard orders.

Prerequisites

You must only activate the event-receiver linkage for the IDoc interface once, because an **event** is always triggered when an IDoc is received (exception: port type “tRFC“). This takes place in Customizing, activity Activate event-receiver linkage for IDoc inbound processing [Ext.].

Process flow

1. The inbound IDoc is linked to the required processing type via the process code [Page 71] in the partner profiles [Page 68]. You can decide whether a workflow or a function module is triggered when an IDoc is received.

   The process codes supplied with the standard system are already assigned to workflows or function modules. You can display this assignment: from the initial screen of the IDoc interface (transaction WEDI), choose *Control* → *Process code inbound*. This is also the initial screen for new assignments when you want to define new IDoc types or processing types. For more information, see define new IDoc types [Page 153]

   A vendor receives a purchase order for a material via an IDoc of type ORDERS01. The vendor has assigned the function module IDOC_INPUT_ORDERS, which converts the IDoc data to the corresponding application data, to the ORDERS message via the process code ORDE. The vendor, therefore, selects the direct path via a function module.

   There are IDoc types for which inbound processing only takes place in Basis, for example TXTRAW02. These IDoc types are only processed by workflow. The corresponding tasks are grouped together in task group TG70000016. Inbound processing by workflow in logistics is located in task group TG20000011. Task groups make the search for tasks in the *Business Workflow Explorer* (Area menu SWLD) easier.

2. For exception handling in inbound processing you must assign the **possible agents** to the corresponding tasks. You have two alternatives:

   - You must classify all tasks as *general tasks* in IDoc Customizing.
   - You maintain the allocation for each individual task via transaction PFTC. The section on Exception handling: procedure [Page 28] describes which tasks are used.

Inbound Processing: Technical Background

1. The IDocs are received by various function modules or reports, depending on the inbound port. The table below provides an overview and shows the interface (reference fields or structures, **import parameters**). Table parameters (*call by reference*) are marked separately.
2. The function module IDOC_INBOUND_WRITE_TO_DB is responsible for saving the data to the database. The function module also uses the process code to determine whether processing by ALE services is to take place.
3. The IDocs are then transferred to the application function module by IDOC_START_INBOUND. In all cases, except those in which the IDocs are transferred via the tRFC port, a workflow **event** is triggered which, in turn, triggers standard task TS30200090.
4. IDOC_START_INBOUND starts a workflow or a function module. This decision is also controlled via the inbound process code.

Table: Inbound function modules depending on port type

Port type: **Function module/ report**	**Formal parameters: Import,** **export (e), int. table (iT)**	**Reference field/** **Structure**	**Comments**
File: EDI_DATA_INCOMING	path name port	EDI_PATH-PTHNAM EDIPO-PORT	
tRFC: INBOUND_IDOC_PROCESS	idoc_control (iT) idoc_data (iT)	EDI_DC EDI_DD	Function module for short names (before Release 4.0)
tRFC: IDOC_INBOUND_ASYNCHRONOUS	idoc_control_rec_40 idoc_data_rec_40	EDI_DC40 EDI_DD40	Function module, Rel. 4.0 (long names)
CPI-C (R/2 System): RSEINB10	p_myport (Port in R/3 Sys.)	EDIIO-PORT	Report to be scheduled as job

Internet: SX_OBJECT_RECEIVE (SAP*connect*)	receive_info document_data system_data receivers (iT) packing_list (iT) object_header (iT) contents_bin (iT) contents_txt (iT) object_para (iT) object_parb (iT)	SXRECINFI1 SXDOCCHGI1 SXSYSDATI1 SXEXTRECI1 SXPCKLSTI1 SOLISTI1 SXLISTI1 SOLISTI1 SOPARAI1 SOPARBI2	Info about sender Document properties Recipient Info about data tables Special header Binary document content ASCII document content (Set/get parameters) (Fields, values for processing)
Synchronous RFC: IDOC_INBOUND_SYNCHRONOUS	int_edidc online docnum error_before_call_application int_edidd (iT)	EDI_DC EDI_HELP-ONL_OPTION EDIDC-DOCNUM EDI_HELP-ERROR_FLAG EDI_DD	Function module for short names (before Release 4.0)
IDOC_INBOUND_SINGLE	pi_idoc_control_rec_40 pi_do_commit pe_idoc_number (e) pe_error_prior_to_application pt_idoc_data_records_40 (iT)	EDI_DC40 EDI_HELP-DO_COMMIT EDIDC-DOCNUM EDI_HELP-ERROR_FLAG EDI_DD40	Function module for Release 4.0 (long names)

Status Processing

Use

The status records log the stages through which the IDoc passes along its path, for example “generated” or “ready for dispatch”. This data is important for monitoring communication and compiling statistics.

Features

Processing

The receiving system can provide information on the processing stages for an IDoc which has been received The following **status confirmation** (or status report) methods are possible:

- Status records can be forwarded to the IDoc interface via the file port. The IDoc number is used for identification purposes. The receiving system determines at what time the status message is returned. Each status record must contain the number of the IDoc to which it refers. In this way, a link can be created to the IDoc and to the application document in the IDoc interface.

  The status records that the confirmation can contain depends on the configuration and technical capabilities of the subsystem.

- Status records can also be returned as an IDoc (IDoc type SYSTAT01). In this case, all possible communications paths (= port types) are permitted.

  Do not confuse the status information contained in the SYSTAT01 data records, which refer to a different (your) IDoc received by the partner, with the status records of SYSTAT01 itself! These are not discussed here.

Exception Handling

If the status confirmation indicates a communication error, **exception handling** is started (see also: Role Resolution in Exception Handling [Page 30]). A workflow, for example, can be triggered, during which the users responsible can respond to the communication error.

In the standard system, exception handling is implemented via process code EDIR, which refers to standard task TS70008125. As the agent responsible you display the status record of the corresponding outbound IDoc when executing the work item. From there you can attempt to resend the IDoc. The partner profiles, however, are not read again: All data required for transmission is read from the control record. If, for example, you have exchanged the port in the partner profiles, the IDoc is still sent to the original port.

As an alternative there is the process code EDIS with the standard task TS30000078, which only allows a status record to be displayed, but not the resending of the outbound IDoc.

When a Release change occurs, you must carry out the IMG activity *Extended exception handling for status confirmation use*, in order to switch from EDIS to EDIR.

Status Processing

The agents responsible are determined in the following order:

1. If a user is entered in the outbound partner profiles, that person is notified.
2. Otherwise, the person defined in the general partner profiles is notified.

Activities

You can decide which status is classified as an error status and therefore starts exception handling: From the initial screen of the IDoc interface, select *Control* → *Maintain status values*. In the detail screen enter a *Process code* that refers to a workflow. The workflow implements the exception handling in the standard system. In general however, it does not have to be like this. For more information, see Exception handling: Procedure [Page 28]

In status maintenance you can also display or change the **status group** (qualification) assignments which play an important role in the monitoring programs.

Inbound Processing of SYSTAT01

Use

Status confirmations via a port of any type are implemented via inbound processing of IDoc type SYSTAT01.

Prerequisites

You have maintained the Partner profiles [Page 82] for the logical message STATUS. The process code is called STA1.

Activities

Inbound processing takes place via the single-step task TS30000206 provided by SAP. The status records transmitted in the IDoc are credited the outbound IDocs that they refer to.

If a status classified as incorrect has been transmitted in SYSTAT01 (see Status processing [Page 23]), the corresponding exception handling for Status processing [Page 23] follows for the relevant outbound IDoc.

If not all status records could be successfully assigned to the relevant IDocs, then the inbound IDoc of type SYSTAT01 has the status 52. If none of the IDocs transmitted can be successfully processed, the IDoc has the status 51. In both cases, exception handling for the inbound IDoc is started.

Event STATUSIDOCERROR is also triggered, which has single-step task TS30000207 as its receiver. As selected agent you edit the IDoc when executing the work item. You can, for example, change the number of the IDoc to which a specific status record refers and therefore allocate the status to the correct outbound IDoc. You can import the changed IDoc again, if you choose *Edit* → *process* in the display. The system then tries again to process the remaining status records that were not successfully posted. If this is successful, the IDoc is assigned status 53.

If you want to cancel IDoc processing, choose *Edit* → *Process* → *Set delete indicator* from the IDoc display [Page 117]

Exception Handling

Use

If errors occur, **work items** associated with **standard tasks** are generated. These work items appear in the integrated inbox of the selected agent. The standard tasks are identified by **process codes**.

Features

Processing via workflow tasks

The standard tasks for the IDoc interface are divided according to inbound and outbound processing. Formal errors can occur, for example, during inbound processing, if the partner profiles [Page 68] cannot be found. You can also activate the **syntax checks** for IDocs in the partner profiles. An error which occurs during such a syntax check is assigned to a special standard task (see also Exception Handling: Procedure [Page 28]

Exception handling in the IDoc interface is enhanced in inbound processing with exception handling in the application, which can respond to status “51” (application document not posted).

The agent for a work item (that is to say for an instance in the general defined workflow task) can eliminate the error and restart the conversion of the received IDoc or mark the IDoc for deletion.

Permitted agents

In all exception situations where the sender is defined in the partner profiles, the permitted agents are read from the profiles. If agents are entered for the special **logical message**, these are notified, otherwise the agents defined for this partner in the general partner profiles are notified.

In all exception situations where no matching partner profile could be read, the agents are determined from the system parameter table (IDoc administrator [Page 249]). SAP strongly recommends that you enter an agent here.

Agents can be **organizational units** (for example, department, job) and not just SAP users.

In order for the permitted agents to be notified by work item, they must be assigned to the corresponding standard task. For more information, see Role resolution in exception handling [Page 30].

Internal and external error messages

Internal error messages refer to errors in your R/3 System. These error messages or error statuses are associated with **system process codes** (in the IDoc interface). **External error messages** are Status confirmations [Page 23] for errors in the external system. They are associated with **status process codes**. If any internal errors (status 51) are found in the application, the application itself generates the corresponding work item, that is to say, no process codes are used in the IDoc interface here.

Special features of the “file” port type

Files can contain several IDocs. The IDoc interface reads these IDocs separately and stores them in the database. Therefore, if a read error occurs, only the IDocs which have not yet been saved are affected. The point at which the error occurred is marked. If the error cannot be

corrected, the file can be read manually from the point at which the error occurred (see Exception handling: Procedure [Page 28]).

The IDoc interface starts inbound processing as an asynchronous process in the background. As a result, the data can be received quickly and reliably from the external system.

Activities

Exception Handling: Procedure [Page 28]

Error handling is controlled by process codes which refer to standard tasks. This section provides an overview of the standard tasks, as well as examples of possible causes of errors.

Exception Handling: Procedure

Use

The following table provides an overview of the errors defined in the standard system and the tasks which are linked to the errors via the process codes. Possible (but not all!) causes of error are specified to aid troubleshooting.

Error	Process code	Workflow task	Possible causes (examples)
IDoc could not be generated	EDIM	TS30000020	Inbound: Error reading file;Outbound: Error generating IDoc from an MC record
IDoc could not be generated under MC	EDIN: Evaluates the partner data from the MC record	TS70008037	No process code in the additional partner profile MC parameters; Error when writing the application data in the IDoc.
Status file could not be read completely	EDIL	TS70008373	
Error during inbound processing	EDII	TS00008068	Process code does not exist; Error during forwarding to ALE services
Error during outbound processing	EDIO	TS00007989	Error during forwarding via RFC; Error writing file
Error during outbound processing affecting all IDocs in an IDoc stack (“Output mode: Collect IDocs")	EDIP	TS60001307	Error during forwarding via RFC; Error writing file; Port does not exist
IDoc syntax error (outbound; syntax check activated in partner profiles)	EDIX	TS00008070	
IDoc syntax error (inbound; syntax check activated in partner profiles)	EDIY	TS00008074	
Error status reported	EDIS	TS30000078	Conversion error in EDI subsystem

Error status reported	EDIR	TS70008125: In contrast to TS30000078, you can resend the IDoc here	Conversion error in EDI subsystem

The errors are processed from the work items in the integrated inbox. If a **file read error** could be corrected, for example, the IDocs which have not yet been read can be read and stored in the database manually, by starting report program RSEINB00.

Exception handling in the application, which can respond to the status 51 (application document not posted), is similar to exception handling in the IDoc interface.

The exception handling tasks in the IDoc interface are summarized in task group TG70000015, those from logistics applications in group TG20000011. You can display task groups with the *Business Workflow Explorer* (area menu SWLD).

Role Resolution in Exception Handling

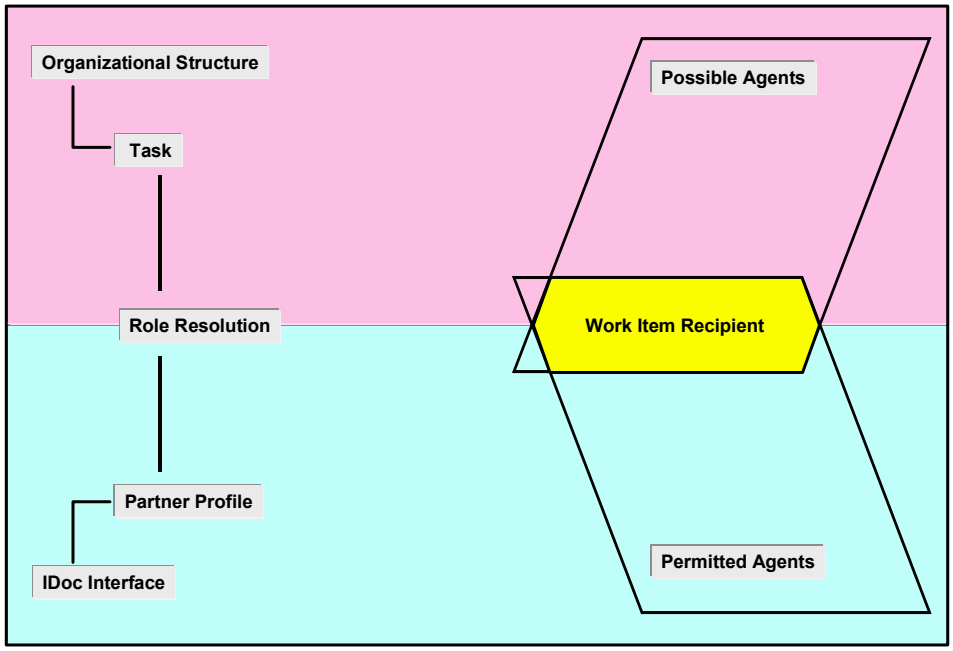
Use

All tasks defined in the standard system for exception handling have a **default role** in order to restrict the number of possible agents responsible for the concrete exception situation. For this purpose, a function module derives agents from the partner profiles for the incorrect IDoc in **role resolution**. If this action fails or if an IDoc does not exist at all yet, the IDoc administrator is determined. Thus, agents with different responsibilities can be determined:

- For the current partner and the current message (outbound- or inbound partner profiles)
- For the current partner (general partner profiles)
- At least for the IDoc interface (IDoc administration)

Integration

The actual agents (the **recipients**) of the work item are the agents determined in role resolution, who at the same time are possible agents of the standard task for the work item. The recipients therefore form the intersection from the possible agents of the appropriate standard task and the “permitted agents“ of the appropriate partner profile or the IDoc administration.



Prerequisites

For receiver determination you must therefore maintain the following:

Partner profiles [Page 74]

IDoc Administration [Page 249]

Agent assignment [Ext.] of the standard task

If the standard tasks are maintained as **general** tasks, the maximum number of possible agents (that is, all users in the R/3 System) is available, with the result that an agent can always be determined (exception: The IDoc administrator is not maintained).

Features

The following table shows the roles supplied with the standard system, with the function module (FM) which is used for role resolution. The table indicates the standard tasks to which the standard roles are assigned and describes what happens during role resolution.

Standard role/FM	Standard role for standard task	Role resolution
Inbound, outbound		
30000013 EDI_ROLE_FOR_ PROCESSING	TS00008068 (inbound processing error), TS00007989 (outbound processing error), TS00008070 (outbound syntax error), TS00008074 (inbound syntax error) TS30000207 (error during inbound processing of SYSTAT01 [Page 25])	The permitted agents are determined from the partner profiles. The IDoc administrator is notified if no agent could be found.
30000001 EDI_ROLE_FOR_ MESSAGE	TS30000020 (error not assigned to IDoc) TS60001307 (error during outbound processing of IDoc stack, that is, output mode “Collect IDocs”)	The IDoc administrator is determined.
70000141 EDI_ROLE_FOR_NAST_ MESSAGE	TS70008037 (error not assigned to IDoc for outbound MC)	If an application problem is involved (for example if an error occurs when writing the application data in the IDoc), the procedure is the same as in function 30000013. If a technical problem is involved (for example if the outbound process code is missing or incorrect), then the IDoc administrator is searched for directly.
Status confirmation		

30000001 EDI_ROLE_FOR_ MESSAGE	TS30000078 (error status in status confirmation) TS70008373 (error during reading of a status file)	The IDoc administrator is determined.
Active Monitoring		
30200013 EDI_ROLE_FOR_ IDOC_ACT_MONIT	TS30200108 (threshold value for active monitoring exceeded)	The permitted agents entered when RSEIDOCM was started or scheduled are determined. If none is found or the agents are not part of the organization model, the IDoc administrator is determined.

Communication with Older Releases/Systems or R/2 Systems

Use

If you wish to communicate via IDocs with earlier R/3 Systems (Rel. 2.1 to 3.1) or with external systems (for example EDI subsystems) based on the same releases, you must inform your system via the **version** in the port definition. This ensures that the correct IDocs record types are sent during outbound processing.

Two problems can arise not only in the case of older R/3 releases, but also in the case of R/2 Systems:

- You are using **long names** for message types, IDoc types or segments. Older releases only support names with up to:

  6 characters (message type)

  8 characters (IDoc type: basic type or extension)

  7 characters (segment)

- You are using **extensions** which have been defined for Release 4.X. Older releases (3.0, 3.1) either use other fields to identify these extended IDoc types internally or do not support extensions at all (Rel. 2.1 & 2.2 and R/2 Systems).

In both cases, you must maintain the **conversion tables** in Customizing. These tables convert the old names into new names. The two cases described above can occur at the same time.

Features

During outbound processing, the system determines the correct IDoc record types via the port version. The old names (in the case of outbound processing) or the new names (in the case of inbound processing) are determined from the conversion tables and if new extensions are being used, the required fields in the control record are maintained (outbound processing).

Activities

The port version is used to determine the release status of the system with which you are communicating. The following versions are used:

- Version 1: Releases 2.1/2.2
- Version 2: Releases 3.0/3.1 and R/2 Systems. As each R/2 System is connected to the R/3 System via the port type “CPI-C” and each CPI-C port is connected to an R/2 System, version 2 is already defined for this port type and cannot be changed in the port definition.
- Version 3: Release 4.X (default value)

Maintain the corresponding conversion tables in Customizing, according to the problems described above.

See also:

Long Names [Page 34]

Using Extensions [Page 35]

Long Names

Use

The **extended namespace** is supported since Release 4.0. This concept includes the use of prefixes and long names. The following objects in the IDoc interface are affected by long names:

Object	Length in 4.0 (characters)	Length before 4.0 (char.)
Message type	30	6
IDoc type (basic type or extension)	30	8
Segment	27	7

Each object has a conversion table which is maintained in Customizing.

Activities

Determine which objects you want to exchange with older releases or an R/2 System and check whether these include objects with long names which are not supported by these releases. You should then maintain the corresponding Customizing tables.

If you have defined **extensions** in Release 4.X and want to use long names, you must maintain the corresponding conversion table for extensions in Customizing. Alternatively, you can define extensions with short names to be used only for exchanging data with older releases.

Using Extensions

Use

In older releases, different methods were used to define IDoc types. The differences with regard to extensions, in particular, are greater and must be taken into account during processing.

- Releases 2.1/2.2

  These releases did not feature extensions for basic types, that is to say each IDoc type (then referred to as intermediate structure) was also a basic type and was identified by the DOCTYP field.

- Releases 3.0/3.1

  These releases used extensions (then referred to as extension types) for the first time. These extensions were combined with basic types (then referred to as basic IDoc types) to form new IDoc types. The IDOCTYP (basic type), CIMTYP (extension) and DOCTYP (IDoc type) fields were used for identification.

- R/2 System

  An R/2 System can work with the control records from R/3 Releases 3.0/3.1, that is to say the system recognizes the fields IDOCTYP, CIMTYP and DOCTYP. However, the R/2 System does not support extensions and identifies the IDoc type via only the DOCTYP field.

From Release 4.0 the IDoc type is identified via the IDOCTYP and CIMTYP fields and no longer via the DOCTYP field. Therefore, the different identification fields must be assigned to each other so that new extensions can be used for communication with older releases.

Features

Outbound Processing

The outbound processing module derives the DOCTYP field from IDOCTYP and CIMTYP in a conversion table. If only one basic type is used, the values for DOCTYP and IDOCTYP are the same.

The port set in the partner profile determines the release to which the IDoc is sent (via the **version** in the port definition). The (release-specific) format (the “record types”) for the IDocs is derived from these entries.

An **error** (with subsequent exception handling) occurs if the DOCTYP field is not known:

- Releases 2.1/2.2 (Port version 1) and R/2 System (Port type “CPI-C“)

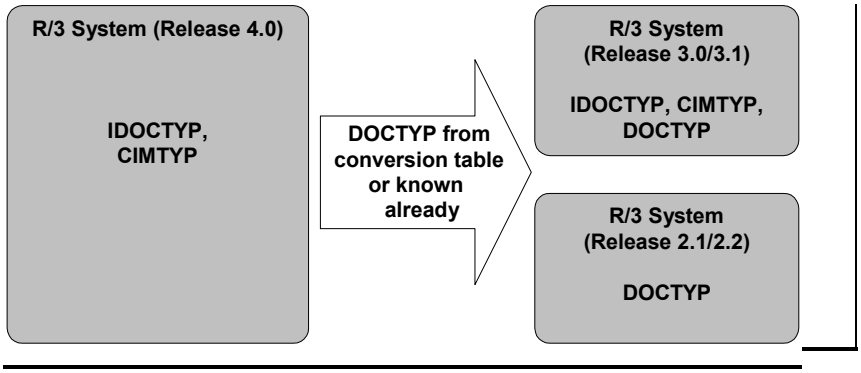
  If the basic type does not correspond to the previous IDoc type (intermediate structure), processing is interrupted with an error. A combination of a basic type and an extension is highly unlikely to correspond exactly to a former IDoc type - if this is the case, however, this assignment must be defined in the conversion table to prevent an error being returned.

- Releases 3.0/3.1 or R/2 System (Port version 2)

  If the basic type and extension are defined but do not correspond to the previous IDoc type, processing is interrupted with an error.

Using Extensions

Assignment of fields in outbound processing



Inbound processing

If the field values are not present, the system tries to determine the values for fields IDOCTYP, and if necessary CIMTYP, from DOCTYP. This can be done as follows:

- If an entry exists in the conversion table, the entry is used to determine IDOCTYP and, if necessary, CIMTYP.
- If no entry is found, IDOCTYP is set to the same value as DOCTYP. This is the case, in particular, when IDocs are received from an R/3 System with Release 2.1/2.2.

An error always occurs if the basic type used (as identified from the IDOCTYP field) is not defined. Errors can also occur if the extension (CIMTYP field) cannot be combined with the basic type.

Activities

IDOCTYP and CIMTYP must be converted to DOCTYP in the following cases:

- You are communicating with an R/3 System with Release 3.0/3.1.
- You or your business partner are using an external system (for example an EDI subsystem) with an older release status.
- You are communicating with an R/3 System with Release 2.1/2.2 or an R/2 System which has defined and therefore “recognizes” your extended IDoc type (as a former intermediate structure). This case is extremely rare.

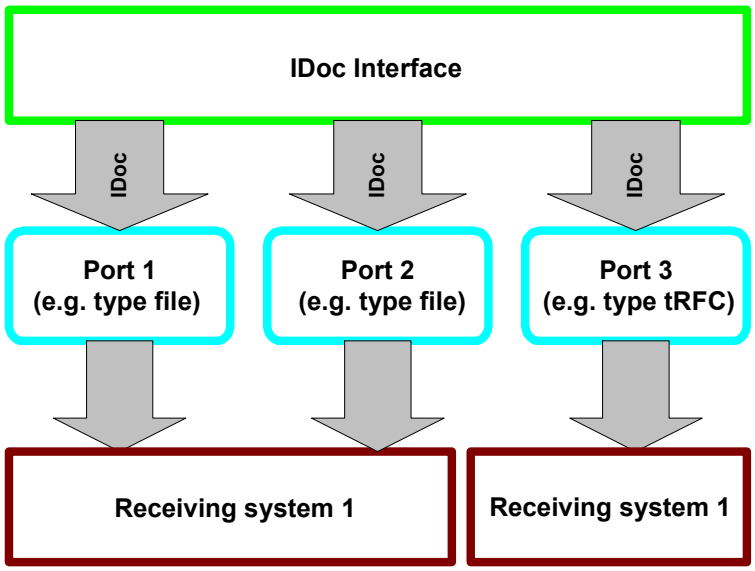
You can convert the fields in the IDoc type editor with which you have defined your 4.X extension (*Environment → Conversion → IDoc type*). You can also use IDoc interface Customizing for the conversion.

Please note that an R/2 System expects IDoc record types from R/3 Releases 3.0/3.1 (port version 2), but in relation to extensions when sending to an R/2 System, your R/3 System behaves as if IDocs are being sent to an R/3 System with Release 2.1/2.2 (port version 1).

Configuring Ports

Purpose

Ports [Ext.] are a fundamental requirement for communicating by means of the the IDoc interface. At least one port must exist for each external system. The figure below shows how IDocs are sent to two subsystems via three ports.



Via the **port version** you inform the R/3 System of the release status of the receiving system. Also read the section

Communication with Older Releases/Subsystems [Page 33]

Prerequisites

You must also make settings outside the IDoc interface so that the port can be used. These settings depend on the required Port type [Ext.] and are specified in the relevant subsections.

Process flow

The process flow depends on the port type. The table below contains application areas and refers to the detail descriptions.

Port type	Application area
File Interface [Page 39]	Link to most EDI subsystems
Transactional RFC [Ext.]	ALE distribution scenarios
CPI-C [Page 51]	Link to R/2 System: Direct communication with an R/2 System (from version 5.0F onwards) is only possible via this port type.
Internet [Page 59]	Sending to e-mail addresses

Programming Interface (PI) [Page 65]	The IDoc is not exchanged with an external system but rather with a function module you have written. This means that any dispatch type can be used.
XML [Page 66]	E-commerce

File Interface

Definition

Exchange of IDocs and status records via operating system files

Structure

Process flow

The sending system writes a file in the file system. Then it notifies the receiving system via the synchronous RFC,

- that the file has been transferred,
- where the file is located (directory) and
- what the file is called (file name).

This type of notification is referred to as **triggering**. For more information see

Outbound: Triggering the receiving system [Page 41]

Inbound: Triggering the SAP System [Page 43]

Inbound processing of status files: Triggering the SAP System [Page 46]

So that no data is lost, the sending system never attaches other data to a file that has already been transferred to the receiving system. The receiving system processes the transferred files completely. It is then automatically deleted from the file system.

To avoid overwriting files, you should use unique names. The SAP System can, for example, name IDoc files according to client and IDoc number.

The sending and receiving system must have the corresponding access authorizations [Page 40] (read and write authorization).

Settings

Port Type File: Maintaining Port Definition [Page 49]

File Interface: Preparing Operating System

1. In the operating system, create read and write authorization for the directories used.

   In outbound processing these are the directories from the Port definition [Page 49], and the authorizations apply to the owner of the SAP Gateway processes, usually the SAP system administrator.

   If the SAP Gateway runs on the same machine as the external system, you do not need to perform the subsequent steps.

2. If the SAP Gateway and the external system run on different machines, they must allow access to one another. With Unix operating systems, this is done via an entry in the *.rhosts* files.

3. Then test the *remote shell* function

   The RFC access is based on this function.

   The external system runs on the machine "marvin". The SAP System C11 with the gateway runs on the machine "arthur". Both machines run under UNIX. "C11adm“ is the UNIX user, to which the gateway process belongs.

   You also configure "C11adm" of "marvin“ as the user, with the home directory /usr/sap/C11/home. Both home directories must contain a ‘rhosts’ file with the following content:

Content of ".rhosts" on "marvin" in ~C11adm	`arthur`
Content of ".rhosts" on "arthur" in ~C11adm	`marvin`

   Both files must belong to user 'C11adm' and have read and write authorizations.

   The entries 'marvin' and 'arthur' in the /etc/hosts files must be identical on both machines.

   You test the *remote shell* function with the following entries:

Machine	Input	Output
marvin	`rsh arthur hostname` (or `remsh arthur hostname`)	arthur
arthur	`rsh marvin hostname` (or `remsh marvin hostname`)	marvin

Outbound: Triggering the Receiving System

Prerequisites

You require the *rfcexec* program from the RFC library.

Procedure

1. To create an RFC logical destination select *Tools* → *Administration* → *Administration* → *Network* → *RFC destinations* (transaction SM59). Position the cursor on *TCP/IP connections* and select *Edit* → *Create*.
2. Enter the name of the RFC destination (for example EDI) and the connection type **t** (start an external program via TCP/IP). Enter a description of the RFC destination. Save your entries.
3. Work without registration: Choose Explicit host or Application server. Specify the program rfcexec (rfcexec.exe for Windows NT) with the directory (for example /usr/sap/<SID>/SYS/exe/dbg/rfcexec). If you have chosen *Explicit host*, enter the target machine on which *rfcexec* is to be addressed (for example `marvin`).

   Work with Registration: Register your program here in the gateway using a freely definable name. Then you can use this ID to call the program with this exact name from the distributed systems. Select registration and enter a name under *Program ID*. Choose *Destination* → *Gateway Options* and enter gateway host and gateway service. You determine these two parameters, for example, with the report RSPARAM (see also Inbound: Triggering the R/3 System [Page 43])
4. Save your entries. Test your destination via *test connections*.

   The connection will only work in registration mode if you have started the program before in the operating system with the same name in the corresponding gateway. The command line for the gateway service **sapgw00** on the gateway host **arthur** is as follows:`rfcexec -a Heribert -g arthur -x sapgw00`,if the program is to be registered under the name "Heribert“.

Result

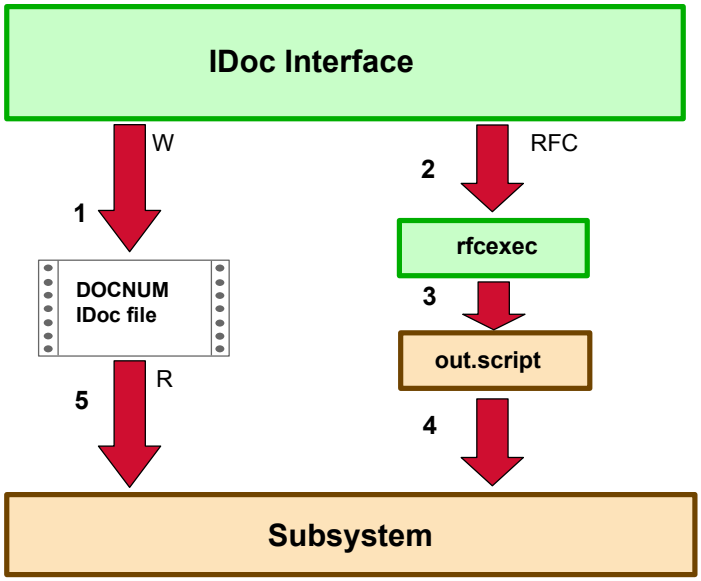
The logical destination specifies the machine on which the **rfcexec** program is to be started. **rfcexec** calls the command file on the same computer. Store the name of this command file, as well as the directory and name of the IDocs to be transferred, in the port definition [Page 49].

Once you have maintained the port definition and have selected *Start subsystem* in the corresponding Partner profiles [Page 206], the process with triggering appears as follows (see the graphic below):

1. Outbound IDocs are generated in the SAP System and written to a file.
2. The SAP System starts the **rfcexec** program via RFC.
3. **rfcexec** starts a script (in the graphic: **out.script**), which it transfers as a parameter directory and a name of the new file (a character string).
4. The script starts the receiving system.

Outbound: Triggering the Receiving System

5. The receiving system reads the new file and then deletes it. It is important that the receiving system stores the numbers of transferred IDocs, so that status information can be assigned to IDocs in the SAP System later.



W = Write R = Read RFC = Remote Function Call

Inbound: Triggering the SAP System

In this section, the SAP System is made known to the upstream system (**startrfc** program parameters). Conversely, the upstream system is made known to the SAP System via the port definition [Page 49] .

Prerequisites

You require the **startrfc** program from the RFC library.

Procedure

Enter the command with the **logon** and **function module parameters** specified below. For example, the command line under Unix could begin as follows:

```
startrfc -3 -d C11 -u EDIUSER ...
```

The parameter sequence is not important.

Logon parameters

Parameter	Value (example)	Description
-3		Logon to the R/3 System via RFC
-d	`C11`	SAP System ID. This is displayed in the system bar, for example.
-u	`EDIUSER`	SAP user Enter the name in upper case letters. For security reasons, you should use a CPI-C user, as this user type cannot be used in a dialog. The user requires all of the necessary authorizations.
-p	`PASSWORD`	User password. Enter the password in upper case letters.
-c	`001`	SAP client (default: 000) as contained in the MANDT field in the control record.
-l	`DE`	SAP system language (default: EN)
-h	`arthur`	Application server (*message server*)Call transaction SM51. The first part of the *server name* field (separated by underscores) stands for the application server. Note upper and lower case letters.
-s	`00`	Two-digit system ID (TCP/IP service):Call transaction SM51. The third part of the *server name* field (separated by underscores) stands for the system ID.

Inbound: Triggering the SAP System

-g	`arthur`	Gateway host: 1. Call transaction SE38. 2. Enter the report name RSPARAM. 3. Select *Execute* and do not select the field *Display also unsubstituted* (the system variable values are then displayed) 4. Select *Find* (CTRL-F) 5. Enter *rdisp/sna_g* and confirm your entry 6. The name of the gateway host is displayed in the line *rdisp/sna_gateway* (you may have to scroll to the right) Note upper and lower case letters.
-x	`sapgw00`	Gateway service as in /etc/services : 1.-5. Repeat steps 1-5 for the gateway host (parameter -g) 6. The name of the gateway service is displayed in the line *rdisp/sna_gw_service* (you may have to scroll to the right) Note upper and lower case letters.
-t		Writes the log data to the file dev_rfc in the current directory.

Function module parameters for inbound IDocs

Parameter	Value (example)	Description
-F	`EDI_DATA_INCOMING`	Only this function module reads IDoc files!
-E	`PATHNAME=` `/usr/sap/C11/SYS/global/EXT/in/<file name>`	Directory and name of the file which contains the IDoc (max.100 characters)
-E	`PORT=EXT`	Logical name of the external system as defined in the port definition (max. 10 characters)

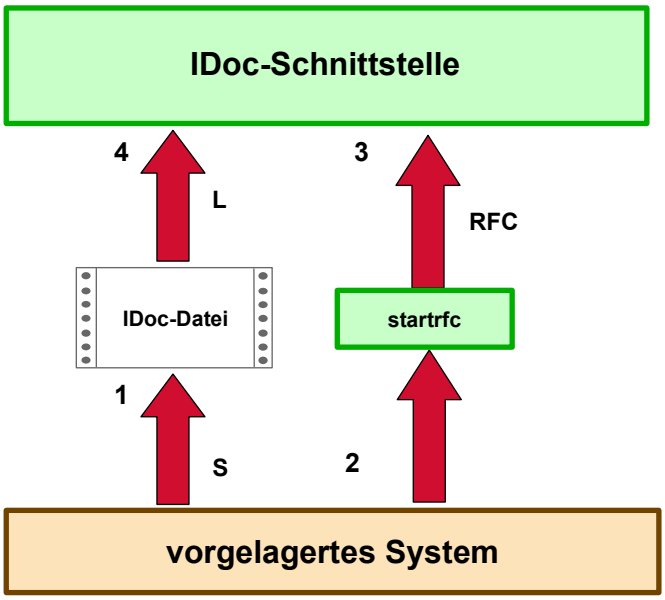
Result

You have notified the upstream system of the R/3 function module to be called and the parameters (file and directory) to be used. Store the name of the upstream system (EXT parameter value in the above table) in the Port definition [Page 49] (section "inbound"), so that the SAP System recognizes the name and accepts the IDocs (see Port type file: Maintain port definition [Page 49] inbound).

Once you have maintained the port definition, the file transfers are triggered in the following sequence (see the graphic below):

1. The upstream external system writes the data to a sequential file in IDoc format.
2. The external system starts the SAP System with the **rfcexec** program.

3. The name and directory of the file (as well as the port) are transferred to the SAP System as **startrfc** parameters.
4. The SAP System reads the inbound file and then deletes it. It does not return any status information to the external system.



S = Schreiben L = Lesen RFC = Remote Function Call

Status Confirmation: Triggering the SAP System

As is the case with inbound processing of IDoc files, the external system must call the program **startrfc**. The status confirmation call differs from the inbound IDoc case only with regard to the **function module parameters**. The **startrfc** must therefore be started with different parameters for the status and inbound IDoc files.

Prerequisites

You require the **startrfc** program from the RFC library.

Procedure

Enter the startrfc command with the **logon** and **function module parameters** listed below. The command line could begin as follows:

```
startrfc -3 -d C11 -u EDIUSER ...
```

Logon parameters

Parameter	Value (example)	Description
-3		Logon to the R/3 System via RFC
-d	`C11`	R/3 System ID. This is displayed in the system bar, for example.
-u	`EDIUSER`	SAP user Enter the name in upper case letters. For security reasons, you should use a CPI-C user, as this user type cannot be used in a dialog. The user requires all of the necessary authorizations.
-p	`PASSWORD`	User password. Enter the password in upper case letters.
-c	`001`	SAP client (default: 000) as contained in the MANDT field in the control record.
-l	`DE`	R/3 System language (default: EN)
-h	`arthur`	Application server (*message server*): Call transaction SM51. The first part of the *server name* field (separated by underscores) stands for the application server. Note upper and lower case letters.
-s	`00`	Two-digit system ID (TCP/IP service): Call transaction SM51. The third part of the *server name* field (separated by underscores) stands for the system ID.

-g	`arthur`	Gateway host: 1. Call transaction SE38. 2. Enter the report name RSPARAM. 3. Select *Execute* and do not select the field *Display also unsubstituted* (the system variable values are then displayed) 4. Select *Find* (CTRL-F) 5. Enter *rdisp/sna_g* and confirm your entry 6. The name of the gateway host is displayed in the line *rdisp/sna_gateway* (you may have to scroll to the right) Note upper and lower case letters.
-x	`sapgw00`	Gateway service as in /etc/services : 1.-5. Repeat steps 1-5 for the gateway host (parameter -g) 6. The name of the gateway service is displayed in the line *rdisp/sna_gw_service* (you may have to scroll to the right) Note upper and lower case letters.
-t		Writes the log data to the file dev_rfc in the current directory.

Function module parameters for status reports

Parameter	Value (example)	Description
-F	`EDI_STATUS_INCOMING`	Only this function module reads status files!
-E	`PATHNAME=` `/usr/sap/C11/SYS/global/EXT/status/<file name>`	Directory and name of the file which contains the status information for outbound messages (max. 100 characters)
-E	`PORT=EXT`	Logical name of the external system as defined in the port definition (max. 10 characters)

Result

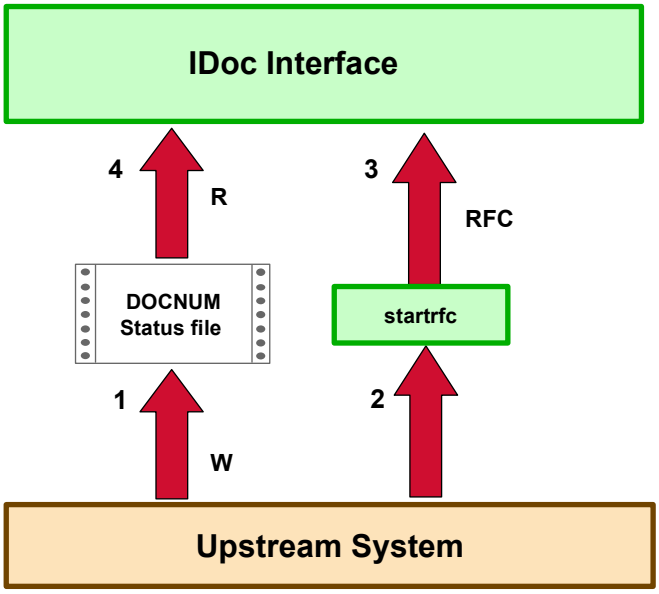
You have notified the upstream system of the R/3 function module to be called and the parameters (file and directory) to be used. Store the logical name of the upstream system (EXT in the above table) in the Port definition [Page 49] (section "status") so that the R/3 System recognizes it and accepts the status records.

Once you have maintained the port definition, the process with triggering appears like the transfer of an IDoc file (see the graphic below):

1. The upstream external system writes the status confirmation to a sequential file in the IDoc status record format.
2. The external system starts the SAP System with the **rfcexec** program.

Status Confirmation: Triggering the SAP System

3. The name and directory of the file (as well as the port) are transferred to the SAP System as **startrfc** parameters.
4. The SAP system reads the status file and then deletes it. The status records must contain the numbers of the outbound IDocs that they refer to (field DOCNUM). In this way, these IDocs can be added to the database.



W = Write R = Read RFC = Remote Function Call

Port Type File: Maintaining Port Definition

Procedure

Outbound

The regulations for generating the outbound file are defined in outbound processing. When you trigger the receiving system via RFC, you also make the command file known (shell script). In the trigger instructions [Page 41] screen it is called **`out.script`**.

The user environment in the operating system is not evaluated. This means that you must set parameters explicitly in the command files and all directories must be specified absolutely.

1. Choose *IDoc* → *Port definition* from the Initial node of the IDoc interface [Ext.].
2. Position the cursor on *File* and choose  .
3. Enter the name of your external system as the port. Enter a *Definition* and *Version* (See F1 Help!) for the record types.
4. Choose *outbound file* and enter a directory.

   Enter a slash (**`/`**) after the directory name, as the path and file are linked.
5. To generate the file names, choose an appropriate *function module* using F4 Help. We strongly recommend that you do not use a static file name, since this file is overwritten each time a new IDoc is sent. SAP recommends that you use the function module EDI_PATH_CREATE_CLIENT_DOCNUM.

   To use “logical directories“ see the F1 Help for the corresponding field.
6. *When you want to trigger:* Choose *Outbound: Trigger* and enter the full directory name for the command file (without the name of the file itself). Enter a slash (**`/`**) after the directory name. You have defined the logical destination in the section Outbound: Triggering the receiving system via RFC [Page 41]. Select *Automatic start possible*, in order to activate triggering.

As well as the  *Access test* in the port definition you can test with the aid of report "Test, RFCEXEC" whether program rfcexec correctly executed the command file. Choose the option RFC_REMOTE_EXEC in the report and enter your RFC destination and your command file as the command (or an explicit command for a pre-test).

Inbound

If the upstream system triggers the SAP System via *startrfc*, the

`-E PORT=`

parameter value of *startrfc* must be entered as the port name. You have already specified the name of the port in step 3 for outbound processing.

Port Type File: Maintaining Port Definition

When the R/3 System is triggered, the name and path of the file are determined by the upstream system. The following settings only provide default values for the test programs that use the file port [Page 102]. Otherwise, they have no effect.

1. Choose *inbound file* and enter a directory as the input parameter for the data.

   Enter a slash (**/**) after the directory name, as the path and file are linked.

2. Enter a file name.

Status files

If the upstream system triggers the SAP System via *startrfc*, the

-E PORT=

parameter value of *startrfc* must be entered as the port name. You have already specified the name of the port in step 3 for outbound processing.

When the R/3 System is triggered, the name and path of the status file are determined by the upstream system. The following settings only provide default values for the test programs that use the file port [Page 99]. Otherwise, they have no effect.

1. Choose *status file* and enter a directory.

   Enter a slash (**/**) after the directory name, as the path and file are linked.

2. Enter a file name.

Configure the following values:

Value...	**for...**
EXT	Name of the external system = port name
/usr/sap/C11/SYS/global/EXT/in	Directory for inbound IDocs
/usr/sap/C11/SYS/global /EXT/out	Directory for outbound IDocs
/usr/sap/C11/SYS/global /EXT/status	Directory for status data

CPI-C Connection to the R/2 System

Definition

Port type for direct access to an R/2 System via the CPI-C protocol.

Use

Since here the R/2 System is always the external system [Page 53], the following exceptions and restrictions apply when compared with other port types:

- The R/3 System is always "active": It either retrieves IDocs from the R/2 System [Page 57], or it sends IDocs to the R/2 System. Therefore in the port definition [Page 55] for outbound processing almost the same parameters are required as in inbound processing.
- The R/3 System can return at most one status record for each inbound IDoc. The IDoc type SYSTAT01 cannot be used because the IDoc type is not available in the R/2 System.
- An R/2 IDoc can only be retrieved once. Consequently, ALE distribution models (IDocs sent to several addressees) can only be implemented indirectly across an R/3 System (the ALE server concept): For this purpose, the R/2 IDoc must be sent first to an R/3 System and then distributed from there to the recipients via an R/3 ALE server.
- IDocs are always exchanged in record types from R/3 Releases 3.0/3.1. This means that port version 2 is always entered by default and cannot be changed in the port definition.
- The R/2 System does not support extensions or the long names used in Release 4.0. You can find out more about the possible problems from the following section: Communication with Older Releases/Subsystems or R/2 Systems [Page 33] .

Structure

Exchanging IDocs between R/2 and R/3

The R/3 System retrieves IDocs from or sends them to R/2 in accordance with the R/2 IDoc interface protocol, that is, R/3 is always the active system. When retrieving IDocs, the R/3 System can define which of the IDocs which are ready for dispatch are actually transferred during the connection. The main obligatory selection criterion is the **port** in the R/2 System. The port is a lock object in the R/2 IDoc interface. You should configure a separate port in the R/2 System for each logical system.

Sending IDoc status records from R/3 to R/2

The R/2 IDoc interface expects a status confirmation for the IDocs which have been sent (that is, retrieved by R/3). You can configure the CPI-C port on the R/3 side in such a way that status records are returned to the R/2 System for the IDocs which are received. You can choose any of the following alternatives for sending status records to the R/2 System:

- do not send status records
- send status records immediately when an IDoc is received
- send the status records asynchronously via a scheduled report

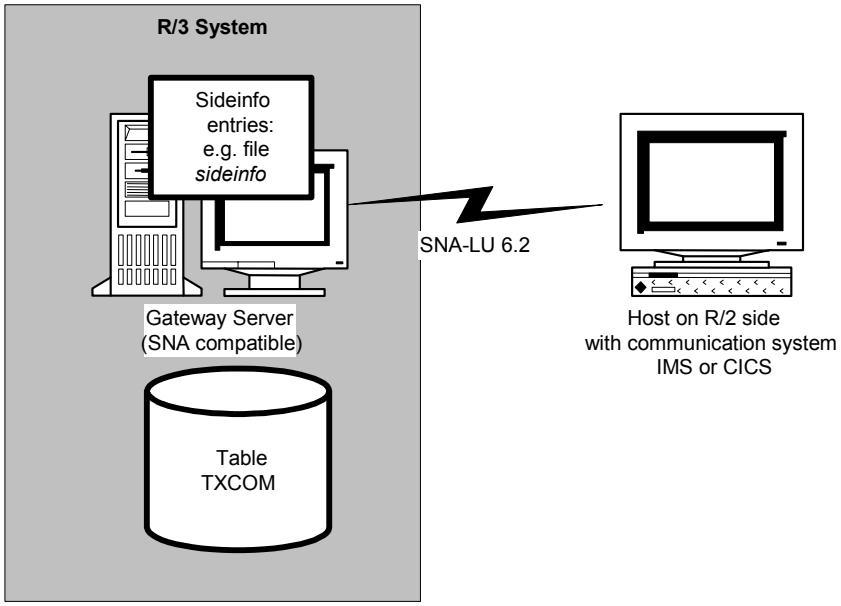
Only one status record can be returned per IDoc (status “12” = “dispatch OK” = “IDoc successfully retrieved by R/3 System”). Confirmation of further processing stages in the R/3

CPI-C Connection to the R/2 System

System cannot be sent to the R/2 System via status records. This can be achieved using response messages modeled in process chains.

Communication technique

The SNA protocol **LU 6.2** (SNA = *System Network Architecture,* on the R/2 side; LU = *Logical Unit*, identifies programs or devices in the SNA) forms the basis for communication. The CPI-C protocol combines the LU 6.2 commands to form a limited number of commands that can be easily used in programming (hence the name CPI-C = *Common User Programming Interface - Communication*). Communication takes place via a **gateway** which recognizes the SNA architecture and assigns the symbolic name of the target system to a *logical unit* in a **sideinfo** file. The name of the gateway and the symbolic name of the target system are located in the R/3 database table TXCOM. In this way, the R/3 System recognizes the R/2 System to be called (see the following graphic, where the gateway is located on the R/3 side).



SNA, LU 6.2 and CPI-C were developed by IBM. The CPI-C protocol used by SAP is a modification of the original IBM CPI-C.

You will find a detailed description of communication with the R/2 IDoc interface in the R/2 manual, S53.2, Chapter 8.

Port Type CPI-C: Linking an R/2 and R/3 System

The R/3 System logs on to the R/2 System to send or retrieve IDocs. The necessary **logon data** is maintained in the R/3 System under an RFC destination. The target system (R/2) and the technical communication attributes are maintained in the table TXCOM.

Maintaining the RFC destination

1. To create an RFC destination, select *Tools* → *Administration* → *Administration* → *Network* → *RFC destinations* (transaction SM59). Position the cursor on *R/2 connections* and select *Create*.

   In the initial node of the IDoc interface go to the same place via *IDoc* → *RFC destination.*

2. Name the RFC destination (for example **`K50-Test`**) and the connection type **2** (connection to R/2). Enter a description of the RFC destination. This data only has to be entered once before you can save the destination. The data is not relevant to the CPI-C connection. The following logon data, however, is important:

   - Client in the R/2 System
   - User and password in the R/2 System

   Save your entries.

   This transaction is called “Maintain RFC Destinations”, but only the logon data (client, user name, password and logon language) is entered for port type CPI-C, not the target system itself. For this reason, the connection test for this transaction does not have to be successful. The actual name of the target system is read as a “symbolic destination” from table TXCOM.

3. If you want the system to make a connection log when sending IDocs, then you must select the field *Trace*.

Maintaining entries in table TXCOM

1. Choose *Tools* → *Administration* → *Administration* → *Network* → *Maintain TXCOM* (transaction SM54). Choose  .

2. Maintain the R/2 System name (for example, **`K50`**) as the destination. Enter the following parameters, in addition to the name:

   - Gateway host and gateway service
   - *Log* `C` (= partner can be reached via CPI-C)

   You can determine the gateway host and service via the gateway monitor [Ext.] or the report RSPARAM.

3. Save your entries.

   The “logical unit” (*LU* column) and the transaction program (*TP* column) are not read from TXCOM, but from the *sideinfo* file on the SNA-compatible gateway server.

Result

The R/3 System recognizes the R/2 System (TXCOM entry) and vice versa (logon data).
Continue with

Configuring the CPI-C Port: Port Definition [Page 55]

Port Type CPI-C: Maintaining Port Definition

Some of the parameters must be configured in accordance with the R/2 communication system (“DC system”) (CICS, IMS under MVS/VSE or UTM under BS2000). Examples of these are provided below:

1. Choose *IDoc* → *Port definition* from the Initial node of the IDoc interface [Ext.].
2. Position the cursor on *CPI-C* and choose  .
3. Enter a name and define the port. Configure the following *CPI-C parameters*:
   - *RFC destination* (for example **`K50 Test`**) with the logon data for the R/2 System
   - *Host destination*

     TXCOM entry with the technical communications data (for example **`K50`**)
   - *Buffer size*

     Length of the CPI-C data buffer (in bytes) which is transmitted with a SEND or RECEIVE command. The maximum value is 8191 bytes.
   - *Number buffer*

     Number of SEND commands transmitted before a RECEIVE command is sent, according to the IDoc CPI-C protocol. A SEND command corresponds to one buffer.
   - *Character string in the R/2 System*. This entry is essential to ensure that the data stream is converted correctly.
   - Transmission modes in accordance with the CPI-C protocol. See the examples provided below. The exact meaning of the following parameters is explained in the R/2 manual, S53.2 (description of the CPI-C interface):
     - Synchronous transmission mode

       If you choose synchronous transmission mode, the *Number buffers* parameter is no longer evaluated.
     - Send acknowledgment of receipt

       The R/2 System must send a confirmation after data transfer. This confirmation does not correspond to a status confirmation, therefore also no new status.
     - Connection terminated by R/2
4. Define an additional option for status confirmation (*status report*) for the R/2 System:

   Specify whether the status records are to be sent to the R/2 System immediately when an IDoc is received, later (that is, scheduled using a report [Page 57]), or not at all.

Result

The transmission parameters are now set. You must still schedule reports for IDoc inbound processing and for subsequent status confirmations for the R/2 System. For more information, please refer to:

CPI-C Port (Inbound): Schedule Reports [Page 57]

Example

CICS data communication system (MVS/VSE operating system)

The selected CPI-C data buffer should be as large as possible (at least 4 KB). The product of the buffer size and the number of buffers must be significantly less than the 32 KB CICS dialog buffer. Therefore a number buffer between 3 and 6 is used (Example: 4 KB × 6 = 24 KB).

For performance reasons, you should not activate the synchronous transmission mode.

IMS data communication system (MVS/VSE operating system)

Depending on the size of the IMS message queue, the length of the CPI-C data buffer should not exceed 1 - 1.5 KB.

Always activate the following transmission options under IMS:

- Synchronous transmission mode
- Send acknowledgment of receipt
- Connection terminated by R/2

UTM data communication system (BS2000 operating system)

The CPI-C data buffer size should be between approximately 5 KB and the maximum value (8191 bytes). There are no constraints concerning the size of the dialog buffer.

Port Type CPI-C (Inbound Processing): Scheduling Reports

Purpose

In inbound IDoc processing, you must schedule a report to select the IDocs in the R/2 System and then transfer them to the R/3 System. If the status “successfully arrived” is not to be returned immediately, an additional report is required. A special report which allows IDocs saved in the R/3 System to be processed by the application in the event of an error is available as a further option.

Process flow

Retrieving IDocs

Schedule the report RSEINB10, which retrieves IDocs from the R/2 System. Enter the following parameters:

- *CPI-C port in R/3 (local)*

  Name of the port type “CPI-C“, which you have just created in the IDoc interface.

- *Output CPI-C trace*

  If the indicator is set, the connection is logged and outputted.

- Additional criteria (for example company code, application transaction or logical message), according to which IDocs in the R/2 System are to be selected.

Delayed return of IDoc status records

If you have specified in the port definition that status records be returned at a later time (that is, not directly after the R/2 IDoc has been received), you must schedule the report RSESTA11, specifying the ports again:

- *CPI-C port in R/3 (local)*

  Name of the port type “CPI-C“, which you have just created in the IDoc interface. The “sender port” field of the IDoc control record (SNDPOR) must contain this name, so that it can be taken into account in the selection.

- *Output CPI-C trace*

  If the indicator is set, the connection is logged and outputted.

IDoc processing after program interruption

IDoc inbound processing essentially comprises two steps. The transferred IDocs are separated from the data stream and stored. The connection to the R/2 System is terminated and the IDocs are transferred to the corresponding application, that is to say the business process which is responsible for the IDoc is started.

If the program is interrupted between the IDoc being posted and processed, the IDocs which have been posted but not processed by the application can be sent. In this case, the report RSEINB11 must be started or scheduled as a job. The only control parameter here is the port maintained in the R/2 System and written to the field RCVPOR in the IDoc control record. All of the IDocs which belong to this port and still have status 50 (IDoc added) are passed to the

application. The corresponding port is determined from the “receiver port“ (RCVPOR) in the IDoc control record.

Port Type Internet

Definition

Conversion of IDocs from or into e-mails.

Use

IDocs can be exchanged via the Internet with any system which has an e-mail address.

Structure

Structure of an e-mail

An IDoc e-mail contains the IDocs as an attachment in MIME format (M*ultipurpose* I*nternet* M*ail* E*xtensions*).

The attachment has the document type R3I and can contain either references to IDocs (IDoc numbers) or IDoc data. The IDoc interface provides two function modules for converting to and from these two formats.

Within the R/3 System, the attachment always contains references to the IDoc data. Outside the R/3 System, the data itself is in the attachment.

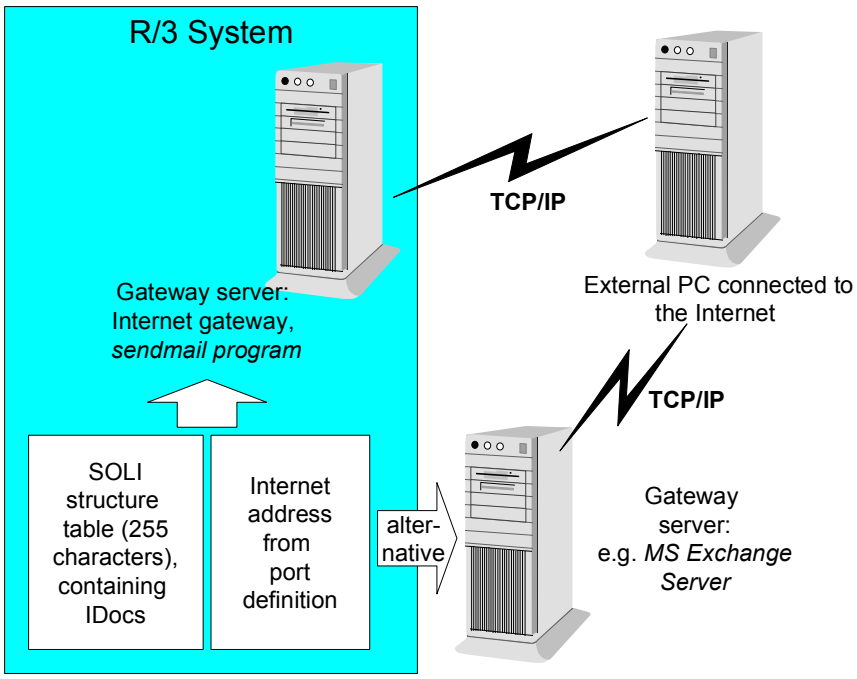
Outbound Processing

A mail, which only carries a set of IDoc numbers as an attachment at first, is generated from the port definition parameters. This e-mail is transferred to the function module SO_OBJECT_SEND, which in turn, determines how the message is to be sent from the Business Workplace.

For that purpose, the Workplace calls the function module EDI_IDOC_R3I_OUTGOING, which returns the IDocs (control record and data records) in R3I format, that is, as a table with the structure SOLI. SAPconnect sends this table to the SAP Internet Mail Gateway, which sends the data via the Internet in MIME format. The gateway uses the program *sendmail*, which is already available under UNIX but must be installed for use with Windows NT (also see the Documentation on the Internet Mail Gateway [Ext.]).

Alternatively, SAP*connect* can activate the *MS Exchange Server* or a different gateway, which is then responsible for sending the data via the Internet.

Port Type Internet



Inbound Processing

The gateway (for example the SAP Internet Mail Gateway) uses RFC to call the SAP*connect* function module SX_OBJECT_RECEIVE, which receives the e-mail in MIME format. The mail is transferred to the Workplace function module SO_DOCUMENT_RECEIVE_API1.

The Workplace calls the function module EDI_IDOC_R3I_INCOMING, which converts the data from R3I format to IDoc format and starts IDoc Inbound Processing [Page 21]. However, it is a prerequisite that the mail recipient of the EDI inbound distribution is assigned (section Configuring the User Address for the Internet [Page 63]).

The IDocs are either generated and processed completely or not at all. For this purpose, EDI_IDOC_R3I_INCOMING is called as a separate work process, so that database changes can be reset separately (rollback) if errors occur during IDoc generation or processing.

As a return value, the Workplace receives a table which contains the IDoc numbers and the name of the **inbound folder** in which the mail should be stored. This folder is determined from the IDoc administration table (section: Configuring an Internet Port: Port Definition [Page 64]).

Integration

For more information on architecture see

SAPconnect [Ext.]

BC - SAP Internet Mail Gateway [Ext.]

SAP Exchange Connector [Ext.] (for the connection to the *MS Exchange Server*)

Port Type Internet: Configuring SAP*connect*

Before using the Internet port, you must define a SAP*connect* node [Ext.] for the address type INT which is to be used for sending mails. The node also controls whether you must specify if the messages are sent via the Microsoft Exchange Server or the Internet Mail Gateway.

The program for retrieving mails, the SAP*connect* Send process [Ext.], must also be scheduled.

Prerequisites

When sending via the SAP Internet Mail Gateway: You must have configured the Gateway. For more information see the documentation Configuring the Internet Mail Gateway [Ext.].

Procedure

1. Call the SAP*connect* Administration by choosing *Tools* → *Business Communication* → *Communication* → *SAPconnect.*

   A tree containing the nodes arranged according to the address type which they support is displayed.

2. If necessary, you can check by choosing *View* → *Routing*, whether a node already exists for the address type INT (Internet).

3. You can change existing nodes via  or insert new ones via  . The following parameters are important:

   - RFC Destination

   Using the RFC destination you define the machine on which the program, either the MS Exchange Server (via the SAP Exchange Connector) or the SAP Internet Mail Gateway, is started.

   - Address area for address type INT

   The more specific this entry, the fewer the addresses are handled via the corresponding node.

   The node KNOT1 supports the address area *sap-ag.de, the node KNOT2 supports the area *C11.sap-ag.de, where C11 is an R/3 System. If an e-mail is sent to <username>@C11.sap-ag.de, KNOT2 is used.

   - Supported format for address type INT

   Here you must specify the document type R3I (Internet IDocs) or * (all formats).

4. Scheduling the Send Process as a Background Job [Ext.].

Port Type Internet: Configuring Addresses for the Internet

These settings are also important for outbound processing: The e-mail sender must be recognized.

Procedure

1. From the R/3 initial screen, choose *Office* → *Workplace* → *Settings* → *User settings*.
2. Choose  .
3. Select *Other communication...* for the communication type INT (Internet).
4. Enter your Internet address in the dialog box, for example **<user>@<company>.com.** Save your entries.
5. Connect your Internet address to inbound distribution: From the initial R/3 screen choose *Tools* → *Business Communication* → *Communication* → *SAPconnect Settings* → *Inbound Distribution* (SO28). Use the F4 Help for the *new recipient* and select *EDI inbound distribution*.

   IDocs which have been sent to you are only processed further through this forwarding.

Port Type Internet: Maintaining Port Definition

Use

The **Internet address** of the partner is stored in the port definition. As the Business Workplace is used here, you can specify the Workplace folder which stores the IDoc mails to be sent as Workplace documents.

Prerequisites

Port Type Internet: Configuring SAPconnect [Page 62]

Port Type Internet: Configuring Addresses for the Internet [Page 63]

Procedure

1. Choose IDoc → *Port definition* from the Initial Node of the IDoc Interface [Ext.].
2. Position the cursor on *Internet* and choose  .
3. Enter a name and define the port. Configure the following parameters:
   - *Internet Address*: Mail recipient
   - *Office Folder:* Workplace folder for outbound mails, mail attributes (title/name) and the sender of the e-mail. This sender is not necessarily identical to the creator of the Workplace document.
   - *Mail text:* The recipient reads this text when they open the mail (IDocs are only attachments)
4. Enter the Workplace folder where inbound mails should be stored in the IDoc Administration [Page 249].

Result

You can now send and receive IDocs via the Internet.

ABAP Programming Interface (PI)

Definition

The exchange of IDocs between the ABAP function modules.

Use

The R/3 System is the “external system” in which further processing can be programmed. Thus, you can for example program your own techniques for transfer to an external system.

Structure

Port Definition

You only have to enter the name of the function module to be called (for IDoc outbound processing) or the calling ABAP function module (for IDoc inbound processing). This function module can be programmed to contain the required processing. Only the interface is already defined.

Interface and processing for IDoc outbound processing

SAP supplies the function module OWN_FUNCTION as a template, which reads the IDoc data and writes a status record for each IDoc (status 18 for “no errors”, 20 for “error”); If an error occurs, a workflow is started.

Interface and processing for IDoc inbound processing

Your module must call the SAP function module IDOC_INBOUND_ASYNCHRONOUS, which saves the IDocs in the database and triggers an event. As a result, inbound processing is started asynchronously.

Port Type XML

Definition

The exchange of IDocs via XML files at the operating system level. The names of XML tags refer to the IDoc record types or IDoc segments.

Use

XML is designed to be a self-descriptive Internet-enabled language. In contrast to HTML, tags can be defined here and recorded in a *Document Type Definition*. As a result, XML is extendible and corresponds in particular to the new definition of IDoc types concept. IDocs in XML format can be displayed immediately with the corresponding Internet browser. *Electronic Commerce* in particular uses XML, but classical EDI applications should also be XML compatible.

Structure

Port Definition [Page 67]

Port Type XML: Maintaining Port Definition

The regulations for generating the outbound file are defined in outbound processing. In inbound processing, the port definition only affects the test program (see below).

Procedure

1. Choose *IDoc* → *Port definition* from the Initial Node of the IDoc Interface [Ext.].
2. Position the cursor on *XML* and choose . Name the port and confirm your entries.
3. Specify a text that is as descriptive as possible in the *definition*.

   The text is displayed in the initial screen of the port definition and in the partner profiles. A descriptive text, for example, would be “files contain the DTD“.
4. Tab page *Outbound File:* You can choose whether the IDocs should be written together with the corresponding *Document Type Declaration* (DTD) in the XML file.

   The DTD contains the tags that are used in the following XML IDoc, that is, tags for the IDoc record types and the individual segments. The tags are named in the same way as the individual fields. The start-end-tag-sequence for the field MESTYP of the control record EDI_DC40 is therefore *<EDIDC_40-MESTYP><EDIDC_40-MESTYP>*.
5. If possible, you must replace country-specific special characters such as ä,ö,ü with international characters like ae,oe,ue. In addition, you must maintain the *Conversion table* and then select *Convert special characters*. You must note, however, that the character strings in the segment fields can then change length!
6. As with port type file [Page 49], it is best if you enter a *function module* which generates the file names. SAP recommends EDI_PATH_CREATE_CLIENT_DOCNUM or EDI_LPATH_CREATE_CLIENT_DOCNUM for *logical directory*. These function modules guarantee clear file names and therefore prevent files that have not been processed from being overwritten. Perform an *access test* for your *directory*.

   The file name generated by the function module overrides the static name, which you can maintain in field *Outbound file*. A static name can make sense for a test port: If your test port is entered in the corresponding field in the IDoc administration [Page 247] , and if you have not entered a function module, then the static name in the test programs [Page 91] is used. Static file names like dynamic file names contain the addition *.xml* in Windows/NT operating systems.

   Logical directories are resolved depending on operating system in the physical directory maintained in the transaction FILE. Also see the F1 Help for the fields *logical/physical directory*.
7. Tab page *Outbound*: *Triggering*: Proceed as in step 6 with Port type file [Page 49]. Perform another *access test*.

Defining Partners

Purpose

You must maintain the partners with whom you communicate via IDocs in the partner profiles: choose the message to be sent to the partner and define the path to be used, as well as how inbound messages are processed.

Prerequisites

- Master data must be available in the system for partners.
- A port is also selected in the outbound partner profiles. This port must be configured already. Also see the section Configuring Ports [Page 37].

Process flow

Standard flow

Purpose of Process Codes [Page 69]

First, read this section in order to familiarize yourself with the concept of process codes as part of partner profiles .

Partner Profiles in the Standard Dialog [Page 74]

Define the messages to be exchanged and the relevant partners. The configuration must always be extended to include new partners and/or messages when necessary.

Checking Partner Profiles [Page 84]

You can check whether the partner profiles you have defined are consistent and complete.

Alternatives to standard flow

Fast Entry in Partner Profiles [Page 85]

You copy default values from Customizing. Individual changes are subsequently possible via the standard dialog.

Programming Interface (API) for the Partner Profiles [Page 86]

You can also define partners automatically via special function modules.

Other functions

Sending partner profiles [Page 89]

Printing partner profiles [Page 90]

Partner profiles cannot be transported. They are maintained in the productive system, after the master data is created.

Purpose of Process Codes

Definition

Another name for a specific process, for example function module or workflow. IDocs are read or written in this process.

Use

In the partner profiles, the processing is never addressed directly but rather always via a process code. You can therefore replace an old process with a new one for any number of partners by assigning the existing process code to the new process.

Two types of process code are used in conjunction with the partner profiles:

- Outbound Process Code [Page 70] - if you are using outbound processing under Message Control, the IDoc is generated in the IDoc interface. The process code names the relevant function module.
- Inbound Process Code [Page 71] - names the function module or workflow which reads the IDoc data and transfers the data to the application document.

If you do not know what process code is relevant for your business process, but you know the message type, then you can display the process code defined for this message type: Choose *Documentation* → *Process codes* in the initial node of the IDoc interface or use the F4 Help in the partner profiles **after** you have entered the message type.

There are also the process codes for exception handling:

- System Process Code [Page 72] - names the workflow which is triggered in inbound or outbound processing when an exception occurs.
- Status Process Code [Page 73] - names the exception workflow which is triggered when an incorrect status is returned by the external system.

These two types are configured centrally and not on a partner-specific basis and therefore do not have to be maintained when a new process is defined. They were introduced for the sake of completeness, so that each process in the IDoc interface is addressed via a process code.

Outbound Process Code

Use

In the case of Outbound processing under Message Control [Page 11], the process is found (function module of the application) via the outbound process code. The process reads the application data and places it in an IDoc.

You must edit the outbound process codes in the following cases:

- You have written a function module (for example for a separate IDoc type), which you want to name through a new process code
- You want to assign a different function module to process code X.
- You want to the switch the ALE services on or off. Deactivating the ALE services can improve performance (lower memory requirement)

Features

The outbound process codes are application specific.

Activities

- You determine the required process code from the corresponding partner profiles or via *Documentation* → *Process code* in the Initial Node of the IDoc Interface [Ext.].
- From the initial node of the IDoc interface, select *Control* → *Outbound process code*.
- Select *Table view* → *Display/Change* to change an assignment or to make a new entry.

Inbound Process Code

Use

The processing (workflow or application function module) which reads the IDoc data and generates the corresponding documents is found via the inbound process code.

You must edit the inbound process codes in the following cases:

- You want to use a new process and need a new process code for it
- You want to assign different processing to process code X
- You want to switch the ALE services on or off (this is only possible if the processing is a workflow!). Switching off the ALE services can improve performance (lower memory requirement)

Features

The inbound process codes are application-specific. IDoc Basis includes the process code ED08, which forwards inbound IDocs to distributed R/3 Systems (“Forward inbound” function). This processing is defined by the workflow template WS30000483.

Activities

- You determine the required process code from the corresponding partner profiles or via *Documentation* → *Process code* in the Initial Node of the IDoc Interface [Ext.].
- From the initial node of the IDoc interface, select *Control* → *Inbound process code*.
- Select *Table view* → *Display/Change* to change an assignment or to make a new entry.

System Process Code

Use

If errors occur in the IDoc interface while IDocs or application data are being processed, a work item is generated. The type of work item (standard task) is dependent on the type of error. The workflow determines how the error is processed from the integrated inbox.

For more information see

Exception Handling [Page 26]

Features

The following process codes are supplied with the standard system:

System process codes

Process code	Description	Standard task
EDII	Error during inbound processing	TS00008068
EDIO	Error during outbound processing	TS00007989
EDIP	Error during outbound processing of IDoc stack (output mode “Collect IDocs”)	TS60001307
EDIX	IDoc syntax error in outbound processing	TS00008070
EDIY	IDoc syntax error in inbound processing	TS00008074
EDIM	Error which could not be assigned to an IDoc (for example, the file could not be opened). No IDoc exists.	TS30000020

Activities

If you want a certain error type to be processed in a different way, assign another task to the corresponding process code.

- You can maintain the system process code from the Initial Node of the IDoc Interface [Ext.] by selecting *Control* → *System process code*.
- Select *Table view* → *Display/Change* to change an assignment or to make a new entry.

You can branch to the workflow definition for the corresponding task from the table by double-clicking on the relevant entry.

Status Process Code

Use

If the external system returns an error status for the IDocs which were sent, an error task is generated.

Features

The following process codes are linked to the following tasks in the standard system:

Status Process Code

Process code	Description	Standard task
EDIS	Error status received from EDI subsystem for outbound IDoc. The corresponding status record and error message are displayed when the work item is executed.	TS30000078
EDIR	As EDIS, apart from when executing the work item you can send the corresponding outbound IDoc again.	TS70008125

Activities

You can use the **status maintenance** function to display or change at which confirmed status the process codes are active:

- From the Initial Node of the IDoc Interface [Ext.] choose *Control* → *Maintain status values*.
- Go into the detailed display of a status. The process code EDIR is stored in the standard system for example for status 07 in frame *effect*.

If you want confirmations of an “error” status to be processed differently, assign another standard task to the corresponding error code.

- You can maintain the status process code from the initial node of the IDoc interface by selecting *Control* → *System process code*.
- Select *Table view* → *Display/Change* to change an assignment or to make a new entry.

You can branch to the workflow definition for the corresponding task from the table by double-clicking on the relevant entry.

Partner Profiles in the Standard Dialog

Purpose

You create partner profiles without default values from Customizing. Although this is the most individual way of maintaining partner profiles, it is also the most complex.

Process flow

Partner profiles are defined separately for inbound and outbound processing. Every partner, however, must be defined in the general partner profiles. For outbound processing under Message Control (MC), you must also configure additional parameters. To define the partner profiles, go to the Initial Node of the IDoc Interface [Ext.].

See also:

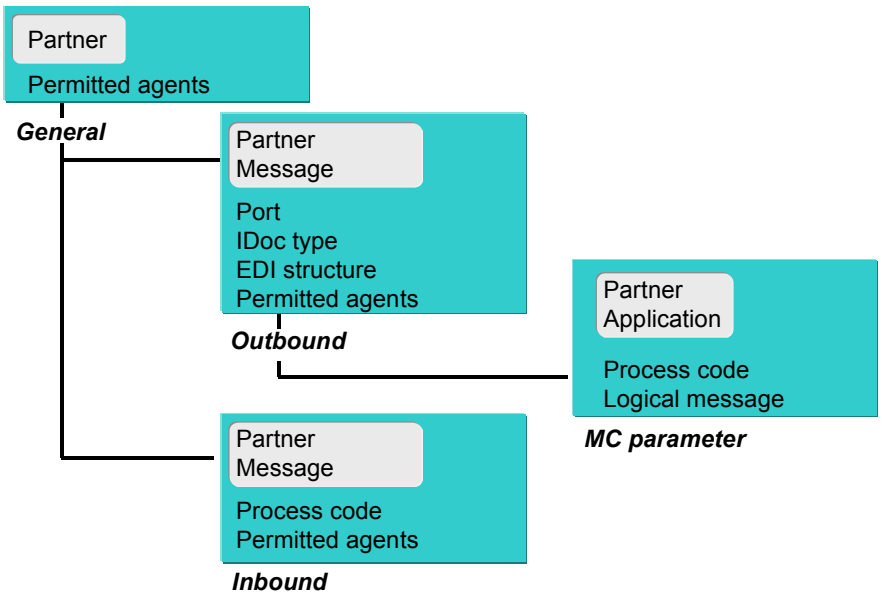
General Partner Profile [Page 75]

Outbound Partner Profile [Page 76]

Additional Parameters for Outbound Processing under MC [Page 79]

Inbound Partner Profile [Page 82]

Graphic: Dependencies and important fields of partner profiles.



The four parts of the partner profiles correspond to the four tables and the keys in the diagram are indicated by lighter fields.

Creating General Partner Profiles

Prerequisites

Data must be available in the system for the partner according to their type (for example, a vendor master record). For more information see Checking partners according to partner type [Page 252].

Procedure

1. From the Initial Node of the IDoc Interface [Ext.], select *IDoc* → *Partner profile* or enter transaction WE20.

   Your partners are divided according to Type [Ext.] (for example vendor). Partner type and partner number clearly identify the partner in the master data.

2. Position the mouse on the partner type required (for example, VE for vendor) and choose  . Further entries are made in the right-hand area of the screen. Enter the partner number here (for example, one of your vendors).

3. You can further divide your partners into **partner classes** (tab page *Classification).* You can then define a new message for all partners in one class using the API (see also: Programming Interface (API) for the Partner Profiles [Page 86]) by creating the message once for the entire class.

4. Set the **partner status** to “active” in order to communicate with the partner (tab page *Classification).* You can deactivate communication with this partner by setting the status to “inactive”. With the “template” status, the partner profile is valid as an (inactive) sample profile for the partner profile API.

5. Define a **permitted agent** for exceptions (for example, if IDocs are transmitted incorrectly)(tab page *Postprocessing: Permitted agent).* For more information on agent determination see Role resolution in exception handling [Page 30]

6. You can use the **archive** indicator to indicate whether documents from this partner have been archived. This indicator is only used for documentation purposes (Tab page *Classification).*

7. If your R/3 System is connected to the telephone system via SAPphone and your computer is assigned to a telephone extension, you can maintain the partner-specific **telephony** data (Tab page *Telephony*). This data can be retrieved from the IDoc Display [Page 117] for the partner, who can then be dialed directly by pressing a button.

Creating Outbound Partner Profile

Here you must enter the data manually. Alternatively, you can also transfer the default values from Customizing [Page 85].

1. If you are not yet in the change screen of your required partner, choose *IDoc → Partner profile* in the Initial node of the IDoc interface [Ext.].

2. Position the mouse on your partner in the required partner type node. Choose  in the *Outbound parameter* table.

Key fields

3. You have already determined partner number and partner type in general partner processing. The partner function from the master data defines the addressee, that is, it is used for further classification purposes. If you have selected outbound processing under Message Control (MC), the function must be identical to the corresponding Message Control field. Otherwise, it is optional.

   Partner A wants to order a material from partner B. Partner B is of the partner type “LI” (vendor) and must choose the Message Control value “VD” (vendor) as the partner function because orders must always be processed via Message Control.

4. Specify the business process with the “logical” message, within which the IDoc type is used. The logical message is described by three parameters: the *message type* is based on EDIFACT message types, for example, a purchase order is of type “ORDERS”. You can further divide the message type with the optional fields *message code* and *message function*.

5. Configure the *test indicator* if you want to send the message as a test.

   Message, partner and test indicator are the seven key fields of the outbound partner profiles (the client comes in addition to these). Also see the figure at the end of this section.

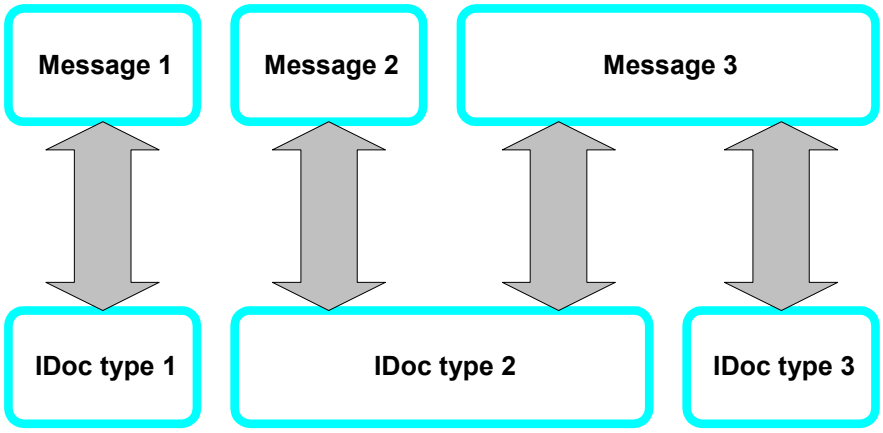
Other fields

6. In the tab page *Outbound options*, you can determine whether IDocs are forwarded immediately to the receiving system. You should ensure that your entries are compatible with the Message Control priorities, if you have chosen outbound processing under Message Control. A list of recommended combinations is provided in the following section: Outbound Processing under M C: Procedure [Page 13].

7. You have already defined the *Recipient port* in Port definition [Page 37].

8. Specify the *IDoc type* as the Basic type with or without extension [Page 155]. If you want to use a view [Page 168] of your IDoc type (for example, to improve the performance), specify this here.

   The figure below shows the m-to-n relationship between logical messages (business meaning) and IDoc types (technical format). Message 1, for example, is always assigned to one IDoc type, while message 3 is assigned to two IDoc types. IDoc type 2, in turn, is also assigned to 2 logical messages.



9. The *segment release* specifies the release from which the segment definitions (not the IDoc type definition) originates. We recommend that you leave this field blank so that the most recent segment definition is used. This field should only be maintained as support information for the EDI subsystem if communication with older R/3 releases takes place via EDI subsystems.

8. You can propose an EDI standard, version and EDI message type for the receiving system in the tab page *EDI Standard*. Most subsystems, however, should be able to determine these EDI settings themselves (from the logical message).

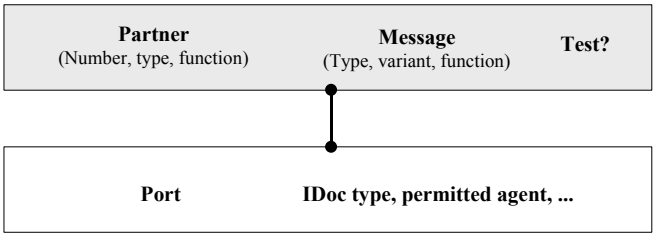
9. You can define *permitted agents* for cases in which exceptions occur. This entry overrides the entry in the general partner profiles. Depending on the message, therefore, the exception can be handled by different agents of the same partner.

10. You can specify whether syntax errors are to be ignored or are to lead to a processing error (*syntax check* in the tab page *outbound options*). For more information about exception handling and permitted agents, refer to the following section: Exception Handling [Page 26]

11. If your hardware supports it, create partner and message specific *telephony* data for outbound IDocs. For more information, read the following section: General Partner Profile [Page 75]

Figure: Outbound partner profile fields (general)

Key fields are shown in gray. The values for partner, message and test indicator (and client) therefore provide a unique ID for the IDoc type in outbound processing.



Additional Parameters for Outbound Processing under MC

This section describes how to define additional outbound parameters, which correspond to the fields from Message default of Message Control [Ext.] (MC). Since the IDoc interface is responsible for generating the IDocs in this case, you must also select a process code.

Prerequisites

- Message Control is available in the system
- You have already created a partner with partner function [Page 76] in the create screen of the outbound partner profile.

  The partner function comes from the master data and is forwarded by Message Control via table MC to the IDoc interface.

Procedure

1. If you are not yet in the change screen with the required partner role, select this in the *Outbound parameter* table. If necessary, choose to display the change mode and choose in the table *Outbound parameter*.
2. Now choose the tab page *Message Control*. For the table, choose .

Key fields

3. Like the partner function, the *Message type* is used to allocate the corresponding condition element [Ext.] of the MC. A purchase order, for example, has the output type “NEU” (new).
4. Enter a name for the application in the column *Application* (for example “EF” for “purchase order”).

   Together with the **change indicator** (for example, for a purchase order change) and the client, the values for partner, output type and application form the key fields of the additional outbound partner profiles under Message Control. Also see the figure at the end of this section.

   The following table contains the most common assignments between Message Control fields (application, partner function and output type) and logical messages in the IDoc interface.

Application	Partner function	Output type	Change	Log. message
EA	VD	NEU (new)		REQOTE
EF	VD	NEU (new)		ORDERS
EF	VD	NEU (new)	X	ORDCHG
V1	SP	AN00		QUOTES
V1	SP	BA00		ORDRSP
V2	SH	LAVA		DESADV

Additional Parameters for Outbound Processing under MC

V3	SH	AUS1		EXPINV
V3	BP	RD00		INVOIC

A message default for MC is always stored as a record in table MC. Therefore, if a message default has already been generated from your application document, you can get the “correct“ application, partner function and output type from this table. Use your document number as the key in transaction SE16.

You must determine in general the partner role of a message [Ext.] (synonym: condition record) in the corresponding application. Thus, you can read this parameter straight from the message definition.

Other fields

5. The *process code* indicates the function module which converts the document into an IDoc.

   F4 Help can be used to display the permitted process codes for the message type (logical message) and IDoc type. A list of all messages with process codes can be found by selecting *Documentation* → *Process codes* from the initial node of the IDoc interface.

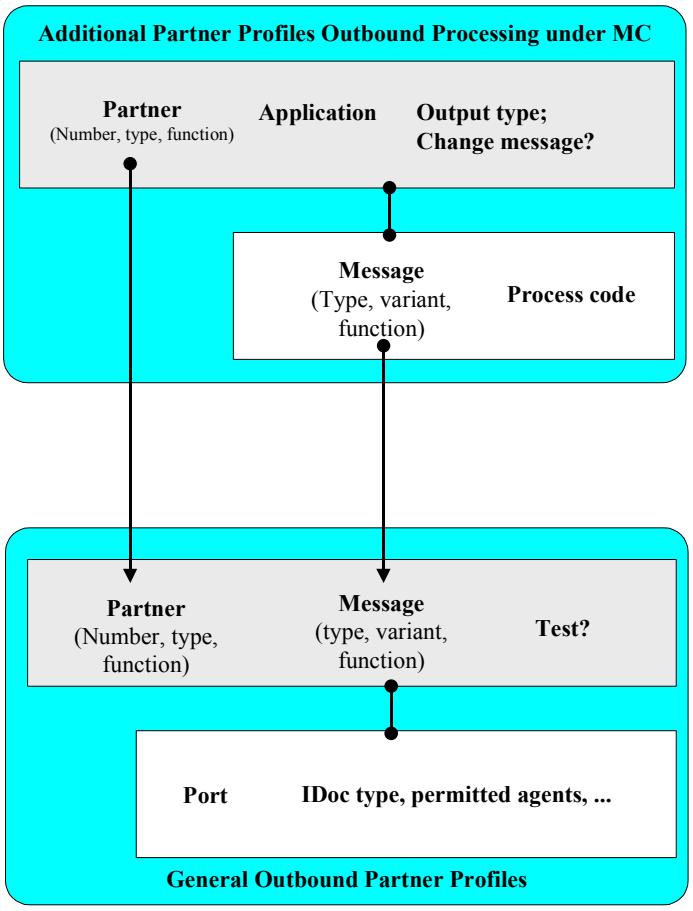
Result

You have assigned the fields in the IDoc interface so that they correspond to the Message Control fields. In addition you have performed an assignment to the general outbound partner profiles (see figure) and indirectly to the general partner profiles.

You should check [Page 84] your settings.

Graphic: Outbound partner profile fields (general and MC specific)

Key fields are shown in gray and the IDoc interface assignments with arrows: partner, applicationand output type (with change indicator) therefore provide a unique ID for the IDoc type in outbound processing (together with the client).


Additional Partner Profiles Outbound Processing under MC
Partner
(Number, type, function)
Application
Output type;
Change message?
Message
(Type, variant, function)
Process code
Partner
(Number, type, function)
Message
(type, variant, function)
Test?
Port
IDoc type, permitted agents, ...
General Outbound Partner Profiles

Inbound Partner Profile

This section describes how to define which messages are to be accepted from which partner. Here you must enter the data manually. Alternatively, you can also transfer the default values from Customizing [Page 85].

Procedure

4. If you are not yet in the change screen of your required partner, choose *IDoc* → *Partner profile* in the initial node of the IDoc interface.
5. Position the mouse on your partner in the required partner type node. Choose  in the *Inbound parameter* table.

Key fields

1. You have already determined partner number and partner type in general partner processing. The partner function is optional.

   Partner B wishes to be able to accept electronic sales orders from partner A. Partner B enters partner A in the partner type “KU” (customer) and can select the partner function “SP” (sold-to party).
2. The message is described by three parameters: The *message type* is based on EDIFACT message types: For example, a purchase order is of type “ORDERS”. You can further subdivide the message type via the message code and function.
3. Together with the *test indicator* (and the client), the values for message and partner form the key fields of the inbound partner profiles. This is identical to (general) outbound processing. Also see the figure at the end of this section.

Other fields

4. The *process code* specifies the process (function module or workflow) which is to be triggered when the IDoc is received (tab page *Inbound options).*

   F4 Help can be used to display the permitted process codes for the message type (logical message) and IDoc type. A list of all messages with process codes can be found by selecting *Documentation* → *Process codes* from the initial node of the IDoc interface.

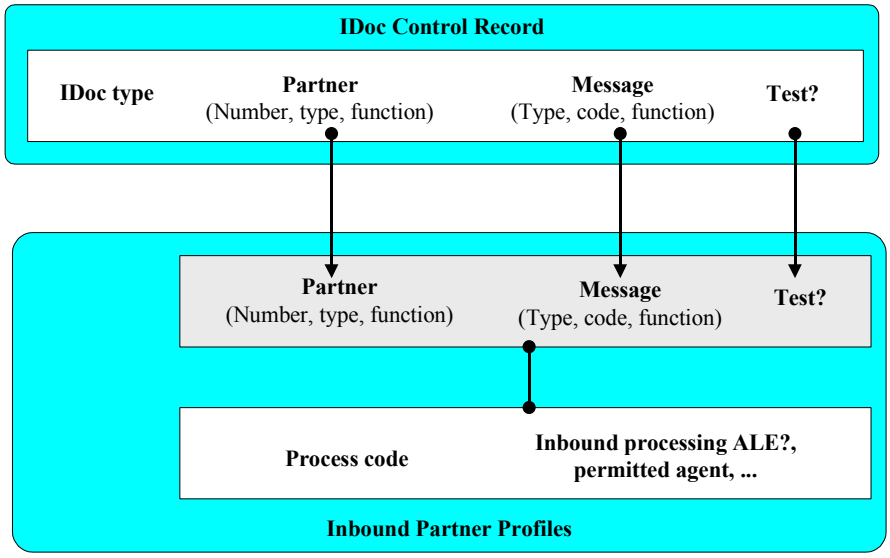
   The current SAP release no longer supports processes. The process technology was replaced by SAP Business Workflow.
5. You can define *permitted agents* for cases in which exceptions occur. This entry overrides the entry in the general partner profiles. Depending on the message, therefore, the exception can be handled by different agents of the same partner.
6. You can decide whether syntax errors in the IDoc are to be ignored or are to lead to exception handling (tab page *inbound options,* field *syntax check*).
7. In the *inbound options* tab page, you can define whether the inbound IDoc is to be processed immediately.

The indicator only affects direct inbound processing [Page 18]. In ALE distribution scenarios for example, inbound IDocs can first be “left idle” and processed later in the background.

8. If your hardware supports it, create partner and message specific *telephony* data for inbound IDocs. For more information, read the following section: General Partner Profile [Page 75]

Graphic: Inbound partner profile fields

Key fields are shown in gray and the IDoc interface assignments with arrows: Partner, message and test indicator (and the client) for an inbound IDoc therefore provide a unique ID for both the process code and the type of inbound processing.



Checking Partner Profiles

Use

You can check the entries in the partner profiles automatically. For example, the system checks whether the permitted agents and the process codes exist.

Features

The entries are checked for each partner.

Activities

- Choose *IDoc* → *Port Profile* from the Initial node of the IDoc interface [Ext.].
- Position the cursor on your partner (identified by the partner number and partner type).
- Choose  and  in the following screen. A list of the check activities and their results is displayed. If errors are found in the partner profile, a message indicating the source of the error is displayed.

  The system always checks whether the additional outbound parameters for Message Control (partner function, application...)

  - exist
  - correspond with those from the corresponding outbound partner profile

  You can therefore ignore these warnings if you use direct outbound processing (without Message Control).

Fast Entry in Partner Profiles

Use

Fast entry reduces the number of parameters which must be set in the partner profiles. Default values from Customizing are used for the remaining parameters.

For an explanation of the individual parameters, read the following section:

Partner Profiles in the Standard Dialog [Page 74]

Prerequisites

You must have configured the default values. They are defined in accordance with the **direction** and **partner type**. You can configure default values in the two following ways:

- In IDoc interface Customizing under *Basis* → *Basis Services* → *IDoc interface Basis*. The corresponding IMG activity is *Set default values for partner profiles*.
- From the Initial Node of the IDoc Interface [Ext.] under *Control* → *Partner profile.*

For the procedure, see the IMG documentation Set default values for partner profiles [Ext.]

Activities

- Choose *IDoc* → *Partner Profile* from the initial node of the IDoc interface. You can select a partner that has already been created.
- Choose  . If you did not select a partner, enter your new partner with number and type here. You should always select the direction.
- A list of preconfigured default values in Customizing is available, arranged according to the key field **Message type**. Message types which are already entered by your partner for the required direction are hidden.
- After you have selected the required message type, the corresponding default values (process code, permitted agents and so on) are transferred for the partner.
- You can adjust individual parameters via the partner profiles in the standard dialog.

Interface (API) for Partner Profiles

Use

You can use the IDoc Interface function modules to maintain the partner data automatically. For example, you can add a large number of partners to the database without having to enter the partner data in the system.

Features

The function modules access the four **partner profiles tables**: EDPP1 (general partner profile), EDP12 (Message Control), EDP13 (outbound parameters) and EDP21 (inbound parameters).

The names of the function modules start with EDI_AGREE. The end of the name describes the relevant database operation.

Function modules for EDPP1 (general partner profile)

Name	Description
EDI_AGREE_PARTNER_INSERT	Inserts new partners. after checking whether: • Partner number exists in the system (for example, master record)? • Partner type is permitted? • Partner status is permitted (“A”, ”I”, or “T”)? • Agent type is permitted? • Permitted agents exist? • Test and archive indicators are set? The partner type and number can be copied without being checked.
EDI_AGREE_PARTNER_UPDATE	Changes partners. Same checks as EDI_AGREE_PARTNER_INSERT.
EDI_AGREE_PARTNER_DELETE	Deletes partners
EDI_AGREE_PARTNER_READ	Reads partner data

Function modules for EDP12 (Message Control)

Name	Description

EDI_AGREE_OUT_IDOC_INSERT	Inserts a new entry. after checking whether: • Partner exists in table EDPP1? • Partner function exists in table TPAR? • Application for message conditions and output type exist as a combination? • Change indicators and test indicators are configured? • Outbound process code exists in table TEDE1? • Logical message type exists in table EDMSG?
EDI_AGREE_OUT_IDOC_UPDATE	Changes a line. Same checks as EDI_AGREE_OUT_IDOC_INSERT, exception: Does not check whether the partner exists in EDPP1.
EDI_AGREE_OUT_IDOC_DELETE	Deletes a line.
EDI_AGREE_OUT_IDOC_READ	Reads a line.

Function modules for EDP13 (outbound parameters)

Name	Description
EDI_AGREE_OUT_MESSTYPE_INSERT	Inserts a new entry. after checking whether: • Partner exists in table EDPP1? • Partner function exists in table TPAR? • Logical message type exists in table EDMSG? • Test indicator and syntax check are configured? • Agent parameters are valid? • Output mode is valid (value between 1 and 4)? • Port is defined in table EDIPORT? • number of IDocs sent per RFC is greater than 0 (packet size - ALE scenario) • Release for which the segment definitions are to be taken exists
EDI_AGREE_OUT_MESSTYPE_UPDATE	Changes a line. Same checks as EDI_AGREE_OUT_MESSTYPE_INSERT, exception: Does not check whether the partner exists in EDPP1.
EDI_AGREE_OUT_MESSTYPE_DELETE	Deletes a line.
EDI_AGREE_OUT_MESSTYPE_READ	Reads a line.

Function modules for EDP21 (inbound parameters)

Interface (API) for Partner Profiles

Name	Description
EDI_AGREE_IN_MESSTYPE_INSERT	Inserts a new entry. after checking whether: • Partner exists in table EDPP1? • Partner function exists in table TPAR? • Logical message type exists in table EDMSG? • Test indicator and syntax check are configured? • Agent parameters are valid? • Process code exists in table TEDE2? • Processing mode is configured?
EDI_AGREE_IN_MESSTYPE_UPDATE	Changes a line. Same checks as EDI_AGREE_IN_MESSTYPE_INSERT.
EDI_AGREE_IN_MESSTYPE_DELETE	Deletes a line.
EDI_AGREE_IN_MESSTYPE_READ	Reads a line.

The following function modules access several tables in the partner profiles:

Other function modules

Name	Description
EDI_PARTNER_READ_COMPLETE	Reads all partner profiles (EDPP1, EDP12, EDP13, EDP21) for a specific partner.
EDI_PARTNER_DELETE_COMPLETE	Deletes all partner profiles for a specific partner.
EDI_PARTNER_COPY_COMPLETE	Copies all partner profiles of a specific partner.
EDI_PARTNER_APPL_READ_OUT	Reads EDP13 for a specific partner. The partner status must be set to “active”.
EDI_PARTNER_READ_OUTGOING	Reads EDP12 and EDP13 for a specific partner. The partner status must be set to “active”.
EDI_PARTNER_READ_INCOMING	Reads EDP21 for a specific partner. The partner status must be set to “active”.
EDI_PARTNER_READ_USER_OUTGOING	Reads permitted agents from EDP13 or EDPP1.
EDI_PARTNER_READ_USER_INCOMING	Reads permitted agents from EDP21 or EDPP1.
EDI_PARTNER_TEST_OUTGOING	Tests whether a partner is defined for outbound processing.
EDI_PARTNER_TEST_INCOMING	Tests whether a partner is defined for inbound processing.

Sending Partner Profiles

Use

You can send partner profiles to an external system via the IDoc type SYPART01. This IDoc type corresponds in the standard system EDIFACT to the messages PARTIN and IMPDEF.

Prerequisites

As you also use the IDoc interface again in this case, you must maintain the outbound partner profiles [Page 76] for IDoc type SYPART01. The logical message for this is called SYPART. A mapping default for the standard system can be found in the documentation [Page 148] for this IDoc type.

Procedure

1. Go into the partner profiles (Initial Node of the IDoc Interface [Ext.] → *IDoc* → *Partner profiles*).
2. Select a partner for which you want to transport data.
3. Choose  .
4. Enter the type and number of the partner to which you want to send the partner profiles.

Result

The system sends an IDoc of type SYPART01 to the partner that you have selected.

Printing Partner Profiles

1. Select a partner from the partner profiles.
2. Choose  and  in the following screen.

   You can also choose several partners by extending the selection criteria correspondingly (in the following screen choose ).
3. If required, you can expand or close individual nodes in the tree displayed. You can mark a section using the Select icon  , which is then displayed by itself.
4. When you are satisfied with the tree output, choose *System* → *List* → *Print*.

Processing Tests

Use

The test programs allow you to skip certain sections of the processing chain between applications so that errors can be located. However, they can also be used to simulate an entire business process (for example, purchase order on the customer side with posting of the purchase order on the vendor side) in an R/3 System (without any other systems). For this reason, the test programs are an important tool for configuring the IDoc interface and defining new IDoc types.

Features

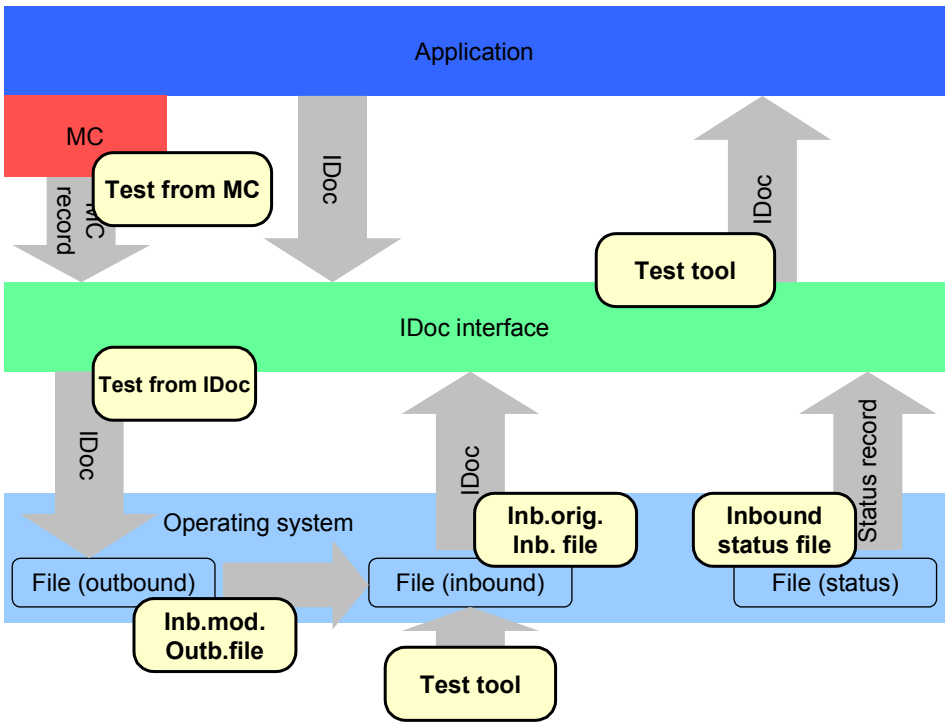
The figure below shows the various test programs and the processing steps when they are used: For example, when using “Test from MC”, an IDoc for outbound processing via the IDoc interface is generated from an existing message status record (which refers to the application data via an object key).

Function	Prerequisite	Result
Test tool [Page 93]	None	One or more inbound or outbound IDocs are generated, depending on the direction, and forwarded for processing.
Outbound Processing from MC [Page 97]	Message status record exists in Message Control, referring to the application object (for example, purchase order)	Outbound IDoc is generated. The IDoc is forwarded to the external system in accordance with the partner profile.
Outbound Processing from IDoc [Page 98]	Outbound IDoc status is 30	IDoc is forwarded to the external system.
Inbound Status File [Page 99]	The status must refer to an existing outbound IDoc.	The status values are appended to the relevant outbound IDoc as status records. Exception handling is triggered in accordance with the status value.
Inbound Processing: Modified Outbound File [Page 101]	File with one or more outbound IDocs exists in the operating system (port type “File”)	IDoc is generated and forwarded for inbound processing (to application)
Inbound Processing: Original Inbound File [Page 102]	File with one or more inbound IDocs exists in operating system (port type “File”)	IDoc is generated and forwarded for inbound processing (to application)

Activities

- The test tool is accessed from the Initial Node of the IDoc Interface [Ext.] in the node *Test*.
- IDocs that were generated with the test tools are indicated with a special status: In inbound processing it is status 74, in outbound processing 42.

Graphic: Interaction of test tools in the processing flow



Test Tool

Use

You can use the test tool to generate an IDoc “manually” and send the IDoc for either inbound or outbound processing. You are not restricted to a specific port type. You can start with an IDoc type (an “empty” IDoc) or use an old IDoc as a template and **edit** the IDoc, that is to say, add segments or change data. This is a good way to test new IDoc types, in particular.

You can forward your new IDoc for standard inbound processing (checking partner profiles and so on). You can also call a function module directly. You can therefore test new function modules for new IDoc types.

Activities

- Start the test tool from the Initial Node of the IDoc Interface [Ext.] by selecting *Test → Test tool.* You can use a template for your test IDoc.

  You can choose IDoc types as a template, either directly or according to a specific message type. You can use the F4 Help for IDocs used as a template, which searches for IDocs by selection criteria, in the same way, for example, to IDoc display [Page 117]. When an IDoc file is used as a template, the IDocs are read from this file and are available to you for selection. A default value for the IDoc file gives you the system via your test port, which you can enter in IDoc administration [Page 247] . This test port must therefore be of the “file“ type. The default file is the inbound file entered there.

- You generate the IDoc via  .

  The IDoc is displayed as a tree structure. If you do not use a template to create the IDoc type, at least one more segment must be added.

- To create segments [Ext.] in the form of tree nodes (colored fields) place the cursor on an existing node (for example a control record at the top) and choose  .

  You can cut, paste or copy individual segments or entire segment groups by positioning the cursor on the relevant segment and selecting the required action from the *Edit* menu.

- Click on the white fields to change data in the segments.

  In the case of the control record, only the fields which are relevant for standard inbound processing are displayed. Do not forget the required entries in the partner profiles if you want to send the IDoc for standard inbound processing! You can also change all of the control record data by choosing *All fields* in the edit screen.

  In the editor screen *All fields* you must enter the **non-language specific** partner function (for example **AG** for vendor). This is the only screen in the IDoc interface in which the partner function is not translated into your language (in English **AG** becomes *vendor* **VD**) - in the partner profiles or in the IDoc display the field is always translated. Thus, you see the partner functions here in the way they are saved in the database. This is a unique value in the R/3 System and therefore protected against mistakes.

Test Tool

- The additional procedure depends on whether you want to test inbound or outbound processing.

  Outbound Test Tool: Procedure [Page 95].

  Inbound Test Tool: Procedure [Page 96].

Outbound Test Tool: Procedure

1. When you have completed your IDoc, choose *Standard outbound*. If you have chosen a valid port, the port and port type are displayed for you to check.
2. Decide if you want to start outbound processing immediately: The IDocs will then be transferred to the port (status 03), *regardless of what is in the partner profiles.* If you do not configure the test indicator, the status of the IDocs remains 30 in the database, again independent of the settings in the partner profiles. You can also send several copies of your IDoc (*repetition factor*).

   If you want to pass the IDocs, which you stored in status 30 with the test tool in the database, on to the port, use Test: Outbound Processing from IDoc [Page 98] The advantage is that the settings then take effect in the partner profiles. In particular, a receiving system is also triggered when the file port is used, if that is defined in the partner profiles.
3. Choose ✔ to start the outbound processing.

Inbound Test Tool: Procedure

1. When you have completed your IDoc, select one of the following **processing options:**

- *Standard inbound processing*:

  The IDoc is treated like an IDoc which was sent by an external system. The IDoc is first saved in the database, then the corresponding control records are compared with the inbound partner profiles and finally the business process determined via the process code is triggered (see Inbound processing [Page 18]).

- *Inbound function module* (inbound processing via ALE):

  the IDoc is first saved in the database and can then be forwarded directly to the specified function module. The function module can be run in debugging mode. If the function module calls a transaction, it can be processed in the background, foreground or in the foreground from the point at which an error is found. In order to do so, however, the ABAP command CALL TRANSACTION must use the import parameter INPUT_METHOD, in conjunction with MODE. Otherwise, the transaction is always processed in the background.

  Function modules called directly must have the ALE interface (Release 3.0 onwards) [Ext.].

- *Inbound File:*

  Generate or extend a sequential inbound file: The IDoc is first saved in the database with status “68” (“error, no further processing”) and then in a file. Specify the name explicitly. The system proposes a name determined from the test port of the IDoc administration [Page 247]. If you select *Start IDoc inbound processing immediately*, the file is processed straight away and then deleted if processing was successful. This mode corresponds exactly to standard IDoc inbound processing. If further processing is not started immediately, the file is kept and can be used for additional test purposes.

  You can write the test IDoc more than once in the file and thus also send them for inbound processing more than once (*repetition factor*).

2. Choose  , in order to start inbound processing from the corresponding screen.

Test: Outbound Processing from MC

Use

Use this test program if you have chosen the Message Control module and want to test generation of an outbound IDoc from an existing message status record (table NAST).

Prerequisites

You must be able to post the application documents which are to be converted into IDocs by the Message Control module correctly so that a message status record can be generated. In the case of the components *Materials Management* (MM) and *Sales and Distribution* (SD) the following entries are required:

- Customer and vendor records
- Material records
- Info records
- MC condition record: The output medium **6** (for EDI) must be entered here. The condition records are maintained as “messages” from the respective application.

The appropriate file ports and partner profiles must exist in the IDoc interface.

Outbound processing must be stopped when the message status record has been generated to allow the test program to intervene. To do so, you must set the Message Control **dispatch time** to “1” (output with selection run) in the corresponding condition record in the application. This test program, therefore, is simply used to start a selection run which retrieves the Message Control records and sends them for further outbound processing. The program is report RSNAST00, which is also generally scheduled with dispatch time “1” in live operation.

Activities

Once the application document has been posted, outbound processing stops after the message status record has been generated and is triggered again by the test program. Choose *Test* → *Test outbound processing from MC* from the Initial Node of the IDoc Interface [Ext.].

Errors are stored in the Message Control processing log (document header) and in the status records of the IDocs. The status records, however, are only available if the IDoc was successfully generated.

Test: Outbound Processing from IDoc

Use

This test program selects one or more outbound IDocs and forwards them to the external system. You can choose the IDocs according to various criteria (for example, recipient or business message).

Prerequisites

You require outbound IDocs which were generated without errors (no error status). The partner profiles, therefore, must be maintained completely.

Outbound processing must stop when the outbound IDocs have been generated, to allow the test program to intervene. You can make sure that this is the case by setting the **output mode** to “Collect IDocs” in the **partner profile** for the IDoc interface. If you now generate an outbound IDoc for the partner (for example, via the application or the test tool), the IDoc is only generated in the R/3 System and is not forwarded to the external system. This test program, therefore, is simply used to start a selection run which retrieves your IDoc(s) and sends them to the external system. The program is report RSEOUT00, which is also generally scheduled with the output mode “Collect IDocs” in live operation.

Activities

Start the test program from the initial node of the IDoc interface by selecting *Test → Outbound from IDoc*.

You can decide whether the output mode is set to “Start subsystem” or “Do not start subsystem” in the partner profile. This defines whether the external (sub) system processes the IDocs further.

Test: Inbound Status File

Use

This program is used to test whether status confirmations for an outbound IDoc are sent correctly from the external system to the R/3 System. The port type here must be set to “File”.

Prerequisites

A correct status file which can be generated by an EDI subsystem, for example, is required. The status file must refer to an existing outbound IDoc in the R/3 System. You can also generate [Page 100] such a status file yourself.

The sender port must be recognized by the receiving system. The port must therefore be maintained as a port of type “File” in the **port definition** for the IDoc interface. The entry for the inbound file must also be specified here.

Features

The R/3 System reads the status file. The IDoc number contained in the file refers to the outbound IDoc, to which the status confirmation relates. The confirmed statuses are “credited” to the relevant IDoc in the form of status records in the R/3 System.

Activities

Start the test program via *Test* → *Process status file* and set the inbound port and the name and directory of the file. These entries overwrite the default values, which you have stored in IDoc administration [Page 247] via the test port.

Generating Status File

Use

For test purposes you can generate a correct status file in the operating system and then send it for status processing. Thus, you can test to see if errors occurred in an external system (that is, if incorrect status files were produced) or in the R/3 System.

Procedure

1. From the initial node of the IDoc interface [Ext.] choose *Test* → *Generate status file*.
2. Specify the number of an outbound IDoc, to which the status confirmation should refer. You can use the F4 Help to search for IDocs according to specific selection criteria, in the same way, for example, for IDoc display [Page 117].

   A choice of status records is displayed, which you can transfer or change.
3. If you want to add a status record, you must choose  and make entries in the individual fields. Via  or  you can copy status records and if required edit them further.
4. For every individual status record you can change the number of the outbound IDoc to which it should refer. Thus, the corresponding IDoc is appended.
5. If you want to change all the fields of a status record, then you must select the record in the table control. Then choose  . The F4 possible entries for IDoc number or permitted status are also available here. Also see the F1 Help for the individual fields!

   In order to identify the “author“ of the status record that was written in the database in status processing, the system enters your user name and the form routine USER_ADDED in program MSEIDOC1 in the status record fields determined for this purpose. You cannot change this information. You can find it again, for example, in the database via IDoc display [Page 117] (tab page *Logging*).
6. When you have edited the data as required, choose  . In the next screen you can still delete the indicator *Start status processing immediately* and thus only generate the status file. Therefore you can view the file in the operating system, which otherwise would be deleted in the course of status processing.

   In order to subsequently process a status file, you must proceed as described in Test: Inbound Status File [Page 99] .

Test: Inbound Processing: Modified Outbound File

Use

This program converts an outbound file with IDocs to a correct inbound file and sends the new file for inbound processing. The outbound file is not modified and can therefore be used more than once. The port type here must be set to “File”.

Prerequisites

You need a correct outbound file, for example, a file which is generated by the test tool or via standard outbound processing. In this case, a port of the type “File” must be specified in the **partner profile** for the IDoc interface, so that the IDoc(s) can be written to a file.

The sender port must be recognized by the receiving system. The port must therefore be maintained as a port of type “File” in the **port definition** for the IDoc interface. The entry for the inbound file must also be given here.

Features

The program imports sender and recipient data as input parameters from the user. The program reads the IDoc file and changes the corresponding entries in the IDoc control record. The changed data is written to a second IDoc file at the operating system level.

Standard inbound processing is then triggered;

- Reading the modified file
- Generating the IDoc(s) in the R/3 System
- Processing in the application

Activities

Set the test program from the initial node of the IDoc interface by selecting *Test → Inb. mod. outb. file*.

Set the sender and recipient data, as well as the outbound file and the inbound file to be generated (path and name). Your entries for the inbound file overwrite the default values which you have stored in IDoc administration [Page 247] .

The recipient in this case is the R/3 System. The port is used for identification purposes.

- SAP<SYSID> (for example SAPC11)

The default values for the partner are:

- partner type US
- no partner function
- Partner number <your SAP user name> (for example PSMITH)

Test: Inbound Processing: Original Inbound File

Use

This program reads an inbound file and sends the file for inbound processing. If all data has been successfully read, the file is deleted.

Prerequisites

You require a correct inbound file. In this case, correct means that the:

- Sender and recipient in the control record are correct
- Direction in the control record is set to 2 (inbound)
- Client in the control record and data records are correct or empty

The sender port must be recognized by the receiving system. The port must therefore be maintained as a port of type “File” in the **port definition** for the IDoc interface. The entry for the inbound file must also be given here.

Features

The program reads the IDoc(s) from the inbound file and sends them for standard inbound processing (with processing within the application).

The file is deleted after being read successfully!

Activities

Start the test program from the initial node of the IDoc interface by selecting *Test → Inb. orig. inb. file* and set the following data:

- Inbound port
- Name and directory of the file

These entries overwrite the default values, which you have stored in IDoc administration [Page 247] via the test port.

Test: Processing Cycle with the CATT

Use

In order to check that IDoc processing worked, you can run a test cycle. Test data is automatically generated, sent as an IDoc and processed further in inbound processing from the IDoc. Thus, you do not guarantee that your specific IDoc type was processed error free with your specific data. You do however verify that in principle, processing worked.

The test is implemented with the Computer Aided Test Tool [Ext.] (CATT) as a test catalog.

Prerequisites

In order to record the result of the test cases in the scope of a test plan, you must next generate one of these. You must assign one or several test packets to the test plan. These test packets are assigned individual persons again, who process the test cases contained within. For more information see

The Test Workbench in the R/3 System [Ext.]

Features

The CATT test catalog contains the following test cases:

- Configuring a port, an inbound- and an outbound partner profile
- Generate an IDoc of type SYIDOC01. The definition of IDoc type TXTRAW01 is transferred in this IDoc.
- Output of IDoc in an outbound file.
- Converting an outbound file to an inbound file and transferring it to inbound processing. Processing of the inbound IDoc is identified via the process code TXT1: The system sends a mail to you which contains the text from the SYIDOC01 IDoc.
- Deletion of the test data, including both test IDocs

The CATT test catalog is also referred to as a (more complex) test case.

Activities

- You start the test from the IDoc administration: In the Initial Node of the IDoc Interface [Ext.] choose *Control* → *IDoc Administration* → *Verification Processing*. The individual test cases are displayed which you can execute individually or together.
- For single execution of a test case, expand the node, position the mouse on the test case and choose . You can also execute the test case on another system, which is identified via a logical destination.
- The execution of a test case generates a CATT process, which is logged. The test log is displayed immediately after the execution of the test case. The process contains a unique number and is deleted as soon as its expiry date is reached. You can view the expiry date by choosing *Edit* → *Process attributes*.
- You can display the attributes of a test case: Choose . You can display the programs that execute the test module within the attributes via .

Test: Processing Cycle with the CATT

- After testing inbound processing, a SAP*office* mail should appear in your integrated inbox. You access this, for example, from the initial R/3 screen via *Office* → *Inbox*. The test is then classified as successful when the IDoc has the status 64 (IDoc ready to be sent to the application).

  Inbound processing is therefore only complete when the IDoc has status 53 (processed successfully in the application). As this part of the processing is triggered asynchronously, the CATT process cannot wait for this status.

You execute the entire test catalog by positioning the mouse on *Test the IDoc interface* and choosing  *Test case*.

For more information see

Executing automatic test cases [Ext.]

Monitoring

Use

You can easily and effectively monitor inbound and outbound processing of IDocs using special reports and graphic displays. An agent can also be notified automatically via a workflow if an “emergency” occurs (**active monitoring**).

Features

The following tools are available for **passive monitoring**:

- IDoc Statistics [Page 106]

  The IDocs are sorted and represented graphically according to predefined status groups. Lists and individual IDocs can be displayed via mouse clicks.

- IDoc Lists [Page 111]

  The IDocs can be selected according to partners and message types. Individual IDocs can be displayed via mouse clicks. IDocs which contain errors are highlighted in a different color.

- IDoc Display [Page 117]

  The selection criteria are extended to include the technical fields of the control record, that is to say, in addition to partners and messages, also

  - IDoc number
  - Ports
  - IDoc types

  You can display a tree structure of the IDoc directly via the IDoc number. The IDoc list is displayed again if several IDocs are selected. This report provides selection criteria which are designed to test EDI subsystems (for example, selection according to the interchange file).

- IDoc Search [Page 133]

  You can select IDocs according to their business content, that is to say, according to the data contained in the segments.

The final section explains

Active Monitoring [Page 121]

IDoc Statistics

Use

The IDocs are grouped together according to their processing status for statistical purposes. The **status groups** shown in the table below are used here. Only those IDocs which have undergone a **status change** within a defined period are selected.

Statistics status groups

Outbound	Inbound
generated	generated
ready for dispatch	transferred to application
being transmitted (transferred to external (sub)system)	transferred to dialog
transmission successful (in target system)	
completed in target system	completed in application
error in IDoc interface	error in IDoc interface
error in external system	error in application
with deletion indicator	with deletion indicator

By double-clicking on individual groups you can see more details: When you double-click on a status group, for example, the IDocs are sorted according to the individual statuses; when you double-click on a status, a list of the associated IDocs is displayed and when you double-click on an individual IDoc, a tree structure of the IDoc is displayed.

You can change [Page 109] the allocation of individual statuses to the status groups.

Features

The following selection criteria are available for statistics:

- Standard evaluation 

  Contains only the time interval for the status changes.

- Extended selection 

  Contains additional restrictions, such as message type, partner and ports. In the selection screen you can also group together the entries according to these criteria (radio button), that is, the statistics are compiled separately for the individual groups.

- Evaluation history 

  Compiles statistics regarding processing of incorrect IDocs by the agent: From the IDocs for which the **current** status was **changed during the respective time interval**, those IDocs which received one of the statuses in the “error” status groups (see table above) **at any time** are selected.

You can also use extended selection criteria here. The IDocs are then subdivided into groups, according to whether the error was corrected or whether the IDocs are marked for deletion (“deletion indicator”).

- ALE audit

  Represents the IDoc data flow between two R/3 Systems in the ALE scenario. For this reason, the statistics are compiled for **outbound processing** in the **local system** and for **inbound processing** in the **target system**; the selection criteria are the logical receiving system and the message type. The IDocs are initially grouped according to individual days! When you double click on a “day bar”, the IDocs are grouped according to their last status, not their status groups.

  Note that the status is also displayed in the R/3 target system (synonyms: "inbound queue" or "receiving system"). Status 64 (“IDoc is ready to be sent to the application”) therefore refers to the application in the target system.

- Selection of test IDocs

  IDocs from the Enjoy Release that are generated from the test tools get a specific status [Page 91]. Only these IDocs are taken into account..

Statistics: Implementation [Page 110] .

Activities

Start the statistics from the Initial Node of the IDoc Interface [Ext.] via *IDoc* → *Statistics*. From here you can branch to the individual evaluations, for example to

Compiling Statistics: Standard Evaluation [Page 108]

Changing segments [Page 109]

Compiling statistics: Standard Evaluation

1. Enter the required date and time (or a date/time interval) in the statistics initial screen.
2. Choose .
3. The required series of values is displayed, which you can represent as a block diagram by selecting *Graphic compl*. To close the graphics window, choose .
4. To display the IDocs in their individual status groups in list form, double-click on the corresponding numeric fields.
5. You can double-click on an individual IDoc in the list to display the IDoc as a tree structure (control record, status record, and segments).

Changing Status Groups

1. From the Initial Node of the IDoc Interface [Ext.] choose *Control → Status maintenance.*

   A list of statuses which have been defined is displayed.

2. Position the cursor on the status which is to be assigned to a different status group and select *Table view → Display/Change*. Double-click on an entry to display the detail screen.
3. The status group is displayed in the “Qualification” field. Use the F4 Help for an explanation of the numbers.
4. Select *Save*.

Status 25 (processing despite syntax error (outbound)) is assigned status group 1 = “Outbound: IDoc generated". Replace 1 with 5 for “Outbound: IDoc interface error".

Statistics: Technical Background

The IDocs are read according to their status values. Since only the last (current) status is relevant for all statistics programs (exception: the evaluation history, evaluation of the test IDoc), the system only accesses the control record table (EDIDC), due to performance reasons - the control record always contains the current IDoc status. Both exceptions mentioned above must read all statuses and therefore access the status record table (EDIDS). For this reason, this program requires more time than the other statistics programs.

The IDocs are assigned to the individual status groups in table TEDS3.

IDoc Lists

Use

This report returns lists, if necessary in graphical format. The selection criteria are

- The time (IDoc creation or last status change)
- The message
- The partners or also the separate identification for the partner
- Technical parameters: IDoc number, IDoc type, current status, direction

The lists are displayed according to message type and status.

You can display statistics for the **time distribution** of the IDocs with a specific status as a graphic. The time axis can be homogeneous (that is, the time progression is uniform) or IDoc-oriented (that is, only time intervals in which IDocs were created are selected).

IDocs with an error status can be easily displayed via the **status bar**. The error message is displayed immediately as a short text and, if necessary, you can also call the long text to receive further information for troubleshooting.

A large number of incorrect IDocs with different message types have been found in the IDoc list for 12/06/1997: The current status of all of these IDocs is 02 (“error while transferring data to port”). IDoc number 4711 is displayed in the status bar and indicates the error number E0099 (“OPEN error <path and file name>“). The long text (double-click) for this error indicates that the <path> directory does not exist or that the entire file system is full. On closer investigation, you discover that there was a file system “overflow” on that day.

Activities

Choose *IDoc* → *IDoc Lists* from the Initial Node of the IDoc Interface [Ext.] .

IDoc Lists: Locating Errors [Page 112]

IDoc Lists: Displaying Time Distribution [Page 113]

Assigning Warning Colors to Status Groups [Page 116]

IDoc Lists: Locating Errors

Errors are shown in the IDoc list with the use of warning colors. The warning colors are assigned specific status groups in the standard system. However, you can also change [Page 116] this allocation.

1. If necessary, change the time interval within which the IDocs were created (default: today). If required, you can restrict the search by using additional selection criteria.
2. Choose .

   The number of IDocs found for each message type and their status in both inbound and outbound processing is displayed. (directions and).
3. Position the cursor on a bar containing an error status (highlighted with warning colors and explained in the text!).
4. Choose a message type from the right window. Choose *Status list.*

   An IDoc list with an error message (number and short text) is displayed.
5. Position the cursor on a specific IDoc. Select *IDoc Display* and from there the corresponding status record. By double-clicking on the short text the corresponding long text is displayed.
6. If necessary, change the allocation of warning colors to status groups.

IDoc Lists: Displaying Time Distribution

1. In the output list, position the cursor on a message (line in the right window) or a node in the left window (all selected IDocs, all in one direction or all with a specific status).
2. Choose *Graphic: Time Distribution*

   A graphical distribution of the IDocs is displayed. The system attempts to define a "sensible" number of horizontal bars:

   - It only displays the times at which IDocs were generated.
   - It searches for the optimal "increments" (for example hours as time steps).

   If you have "outliers" in your time frame, that is, values that are considerably higher than the others, they may be scarcely visible, as the graphic is scaled-up to the maximum level. You can avoid this by choosing a logarithmic instead of a linear scale or by reducing the scale maximum. Select the horizontal scale (axis) with the right mouse key and choose *Format Axis* in the context menu. In the next screen choose the *Scale* tab page and change the settings accordingly.
3. You can change the output form: Choose and in the following window for example *Time step...Day,* in order to set the period pattern to days.
4. To make the time axis homogeneous (each time unit is displayed), select *Display zero values.* In order to display only the time segments within which the IDocs were generated, leave the field blank.

IDoc Lists: Displaying Distribution of Warning Colors

Use

A pie chart is displayed that provides a rough overview of the portion of IDocs with an error status (red warning color). Correspondingly, the other two portions are the green (success status) and yellow (no final status yet) warning colors.

The assignment of warning color statuses takes place in two steps.

1. Assigning status groups [Page 109]
2. Assigning warning colors to status groups [Page 116]

Procedure

1. Enter your selection criteria in the initial screen of the IDoc lists and start the selection.
2. In the output, choose  .

   The overview is displayed in a window below. All IDocs which meet your selection criteria are included in the graphic.

   To hide the window, choose  .

IDoc Lists: Displaying Distribution Of Messages

1. In the left window of the output, position the cursor on the node via which you want to display the distribution:
   - All selected IDocs
   - Only those in one direction
   - Only those with a specific status
2. In the left window of the output, choose the selection arrow on the right next to the  symbol.
3. From the open menu choose *Graphic*: *Message types*.

   A pie chart is displayed that you can further format. If for example you want to show concrete figures, choose the pie chart Format Data Series (right mouse button) in the context menu and in the following screen *Show value* in the tab page *Data Labels*.

Assigning Warning Colors to Status Groups

Use

The current status of an IDoc is indicated with the use of colors in the monitoring programs as either containing errors or successful. You can change which colors are assigned to which status via the allocation of status groups.

Prerequisites

All statuses in a specific status group should have the same color. You allocate the status of a status group in the status maintenance. For further information see change status groups [Page 109].

Procedure

1. From the initial node of the IDoc interface [Ext.] choose *Control* → *Maintain status group.*
2. Select a specific status group (*Qualification*) and choose  .
3. Enter the required *traffic light code* and save your entries.

IDoc Display

Use

This report generates a list of individual IDocs or, if you have restricted your selection to one IDoc using the available selection criteria, the report displays the relevant IDoc (for example, if you select only one IDoc number as a selection criterion).

In addition to the selection criteria, which are also available in the other statistics programs (message, partner), you can select the following, more technical parameters:

- IDoc type (basic type or basic type with extension)
- Test option from the partner profiles (is the IDoc created and sent for test purposes only? – File card *Additional selection*)
- Current status
- Port (recipient and sender)
- EDI reference numbers (interchange file, message, EDI archive – file card *EDI selection*).

  These keys are only relevant for a conversion in EDI standard and refer to objects in the EDI subsystem.

  Do not confuse the interchange file (EDI standard) with the file used by the EDI subsystem to communicate with the IDoc Interface (IDoc standard).

  An EDI partner sends an IDoc to your system for test purposes. The EDI partner is defined as a test partner in your partner profiles. Your EDI subsystem transfers the IDoc to the inbound processing module of the IDoc Interface, where the IDoc is assigned the number 4711. The test indicator is set in its control record. You can now generate a list of all the IDocs with a test indicator via the IDoc display function, in order to locate IDoc 4711.

  In a second test, you wish to determine whether the EDI subsystem has transferred the EDI reference numbers to the R/3 System correctly. You have noted the reference number of the interchange file in which some of the IDocs were transmitted. Enter this number in the corresponding selection field and start the search. All of the IDocs in the R/3 System which were present in the relevant interchange file should be found.

Activities

The IDoc display function can be accessed from the Initial Node of the IDoc Interface [Ext.] by selecting *IDoc* → *Display IDoc*.

Multiple IDocs

If a list of IDocs is displayed, double-click on the corresponding entry to access an individual IDoc.

A traffic light symbolizes whether the current status is a success status or an error status, or whether further processing is necessary, as a decision has still not been made (yellow light). You

can change the color allocation yourself via the transaction WELI. See the *Help on the application* there.

Individual IDocs

General information

The individual IDoc is displayed as a tree structure. The initial node is the IDoc number. The control record, data and status records are displayed as subnodes. You can expand nodes and display individual records. You only have to click on data records.

By choosing Display links [Ext.] from the Object Services [Ext.] you can display the objects that are already linked with the IDoc, for example, application documents or the corresponding IDoc in the partner system in ALE distribution.

Control record

The control record displays, for example, the “letter header” (sender and recipient), direction and IDoc type. You can edit the control record in Exception Handling [Page 26] (*Control record* → *Control/Change*, see also the corresponding paragraph in the data records).

Data records

In the case of the data records, the segment **name** (E1 structures for SAP segments), segment **number** and **short text** are displayed. In the case of qualified segments, the “**qualifier”** is displayed (the value which determines the significance of the segment).

In the case of segment E1EDKA1 (document header partner information), the qualifier PARVW = “SH” means that the partner specified in the segment has the function of the goods recipient, while the partner function of the sold-to party is defined as PARVW = “SP”. The qualifier is the first field in a qualified segment.

If you want to edit an incorrect IDoc in Exception Handling [Page 26], choose *Data record* → *Display/Change* in the detail screen (double-click on  symbol for the corresponding data record). You can now change all the fields. When saving, your IDoc assumes the new status 69 (IDoc edited) and you can then send it for inbound processing again. The original is saved as a new IDoc with the status 70 (original of an IDoc, which was edited) in the database.

Status records

In the case of status records, the individual status values (**status**) are displayed with a **short text**. In the case of Port [Page 37] type "tRFC" and status 03 the unique **transaction ID** which is assigned to the tRFC is displayed. If the application writes an **error log**, the ID of the log is also displayed and can be used to access the log (Goto → *Application log*). This allows you to display more detailed error information than the data in the status records.

IDoc Search

Use

Users should be able to find IDocs not only via the address information or control information in the control record but also according to the business data they contain. You also want to be able to answer questions such as: “Which IDocs contain purchase orders for my material XY?”

The IDoc search function can be used to answer such questions. You can search for IDocs both in the database and in archive files.

Prerequisites

When searching in the archive (as opposed to searching in the database) you must

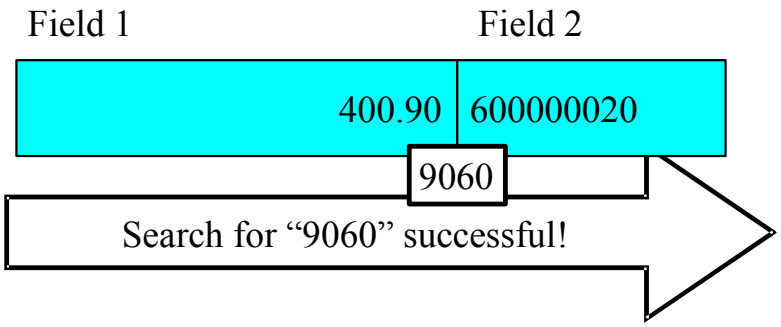
- Have deleted the IDocs from the database (complete archiving session)
- Have created an archive info structure in the central transaction SARA. For further information see Individual display via SAP AS (BC-SRV-EDI) [Page 132]

Features

The function searches for character strings, that is, you must enter the value of a segment field as it appears in the field itself (avoiding full stops and commas whenever possible).

You can enter a maximum of two values in two segment fields as selection criteria. If two values are entered, both must exist in the IDoc in the corresponding segment fields for the IDoc to be included in the list of results (normal AND link in selection fields).

You can also choose to only give the value instead of the segment field. This means that all data records (more precisely: the field SDATA in the data records) are searched for the character string, that is, the character string which is found may span several segment fields.



If you select *Quick search mode*, the search can be limited to a maximum of one hit in each IDoc, that is, the system terminates the search in the current IDoc after one hit.

The search can and should also be restricted by entering additional search criteria which require values from the control record (in the same way as the other monitoring programs): Creation time, partner and message, direction, and so on.

Activities

The IDoc Search can be accessed from the Initial node of the IDoc interface [Ext.] via *IDoc* → *Find IDoc*. Different transactions are called depending on whether you wish to search in the database or in an archive.

Active Monitoring

Use

This report automatically informs the agents responsible when too many incorrect IDocs are found.

Integration

Active monitoring is not used for processing or reimporting of an incorrect IDoc. Exception handling of every incorrect IDoc is responsible for this.

Activities

You plan the report to run regularly with a variant. The IDocs which satisfy the selection criteria are subdivided into status groups for statistical purposes. The variant also informs the report of the **critical status group(s)**. If the critical group contains more IDocs than specified in the **critical number of IDocs**, a message is sent to a predefined recipient.

The recipient receives the notification in the form of a **work item** displayed in their integrated inbox (see also: Role Resolution in Exception Handling [Page 30]). If they execute the work item, the IDoc statistics are displayed with the values determined at the time of evaluation. The agent can display the current status of these IDocs using the *Refresh* function. The selection criteria which led to the notification are also used for this evaluation.

An important customer orders goods via EDI on working days, between 8am and 6pm. These goods are to be delivered at 4pm the next day. To ensure that the delivery is made on time, the ordered quantity must be recorded by midday on the day of delivery. The active monitoring function is to be used at 8am every morning to determine whether there are any orders which could not be processed automatically. The report, therefore, is started at the same time every day and 0 is selected as the critical number of IDocs. The agent responsible is notified if any incorrect IDocs are found and can then manually process the outstanding orders by midday.

See also:

Configure Active Monitoring [Page 122]

Schedule Monitoring Job (example) [Page 123]

Configuring Active Monitoring

Use

You have to configure active monitoring only once: If necessary, you can activate the event-receiver linkage for the task and configure an organizational unit.

Procedure

1. Maintain recipient of notifications in organization model of application component *Organizational Management.* You can maintain a complete organizational unit for notifications. In this case, however, you should check if this organizational unit contains agents! To find out more about maintaining the organizational unit, you should read the section entitled Simple maintenance [Ext.] in the R/3 Library under *HR → PA → Organizational Management → Organizational Structure*.
2. Activate the event linkage for standard task ‘TS30200108’. You must also define who is a possible agent: You can maintain the task as ‘general task’ (every R/3 user is a possible agent), for example. You have the following alternatives for this step:

- Automatic Workflow-Customizing (IMG activity *maintain standard settings for SAP Business Workflow*)
- Direct maintenance of the task: Choose *Tools → Business* Work*flow → Development → Definition tools → Tasks → Change* (transaction PFTC) and choose *triggering events* in order to activate event receiver linkage. Choose *Additional data → Agent assignment → Maintain* to enter possible agents. Alternatively, you can also set an organizational unit as an agent here (for example, the organizational unit which you configured in step 1).

  For more information on possible agents see Role resolution in exception handling [Page 30]

Schedule Monitoring Job (example)

Prerequisites

The following refers to the example in the section Active monitoring [Page 121] . The active monitoring report is to be started as a **batch job** every morning at 8am, in order to evaluate the IDocs which were received between 8am and 6pm on the previous day.

Procedure

1. Choose *Variant* (for the RSEIDOCM report) in the ABAP editor and *Display*.
2. Give your variant a name and choose *Create*.
3. Enter the following parameters:
   - Start time and end time before batch run: `1 Day` and `0 Days 14:00:00h`
   - *Status group*: `F` (inbound: application error)
   - *Critical number of IDocs*: `0`
   - Sender parameter: <corresponding values from the partner profiles of the customer>
   - *Logical message type*: ORDERS
   - *Recipient of notification*: For example `SMITH`
   - *Recipient type*: User
4. Choose *Continue* to enter a short text for your variant. Save your entries.
5. To schedule your job as a batch job, choose *System* → *Services* → *Jobs* → *Job Definition* and enter `B` (medium priority for periodic jobs) as the job class.
6. Select *Start condition* and enter a date and `08:00` as the start time.
7. Choose *Execute job periodically*.
8. Choose *Period values* → *Daily* and save your entries.

   Enter the scheduled report again when saving (RSEIDOCM) and the variant that you maintained.

Result

IDocs selected during the daily batch run will now be:

- those which were received between 8 am and 6 pm on the previous day
- those which have the logical message type ‘ORDERS’
- those received from the relevant customer

An IDoc from status group F must be assigned now to ensure that a notification is sent to the agent SMITH.

Archiving IDocs

Use

IDocs are archived in several tables in the database. In order to limit the size (and thus the access times) of these tables, that is to reduce the load on the database without losing any IDocs, they can also be stored in archives at the operating system level. These archives can then be moved to external storage media, for example optical disks (via SAP ArchiveLink) or magnetic tapes.

Prerequisites

In Customizing you determine the physical location of the archive files, that is, the storage media used. For more information see

CA → Archiving application data → Central Introduction → Customizing → Connection to the archive system [Ext.].

Activities

Displaying or Changing Archivable Statuses [Page 125]

When are IDocs archivable?

Archiving functions for IDocs [Page 126]

How are IDocs written to and read from archive files?

You should ensure that IDocs which may still be required by the application are not archived.

See also:

CA → Archiving application data [Ext.]

Information on archiving in general.

Displaying or Changing Archivable Statuses

Certain IDoc statuses are classified as archivable in the standard system, while others are not. You can display or change this classification.

The **current** status of an IDoc must be archivable before the IDoc can be archived.

Procedure

1. From the Initial Node of the IDoc Interface [Ext.] choose *Control* → *Maintain status values*.
2. Choose  to change the entries.
3. Select the required status. By double-clicking you can call a detail screen, in which you can display or change the archivability.
4. Via  or  you can display or change all statuses one after another.

Archiving Functions for IDocs

Use

Data is usually archived via the central transaction SARA. In the case of IDocs, the corresponding tables are addressed via the **archiving object** IDOC.

Features

Various **processing methods** are assigned to the **archiving class** IDOC: In the standard system, therefore, IDocs can be

- Archived [Page 127]
- Deleted from the database [Page 127]
- Retrieved from an archive [Page 129]: From the initial screen choose *Goto* → *Retrieve*

  All of the IDocs generated by sales orders (message type ORDERS, IDoc inbound processing) in the month of January are to be archived and subsequently deleted from the database tables. To do this, choose the “Archive” processing method in the standard system and enter a time period and message type as report variants.

In addition, you can display the following:

- Archive [Page 130]

  Several IDocs are usually archived in one archive or **archiving session**, that is, the IDocs are grouped together in one session. An archiving session is **complete** when the IDocs have been deleted from the database tables and exist only as archive files.
- The IDoc numbers of one or several archives or archive files [Page 131]
- The IDocs (that is, control record, data- and status records) of one or several archives [Page 132]
- IDocs from archives, that contain certain character strings (IDoc search) [Page 133]

  This function is not an archiving object IDOC method, but rather an IDoc interface monitoring tool and hence only available from the initial node.

Activities

You can also access the central transaction SARA by choosing *Tools* → *Administration* → *Administration* → *Data Archiving*. Here you enter the archiving object IDOC. When you select 'enter' the methods entered above are displayed for selection.

You can determine the Maximum size of the archive files [Ext.] in Object specific Customizing [Ext.] , and the table entries should be deleted immediately after generation of archive files from the database. With large datasets this can improve the performance.

In the standard system, the IDocs are selected for archiving according to certain criteria, for example the time at which the IDocs were last changed (new status record). You can also define your own selection criteria. For more information, refer to the section Archiving: Technical Implementation [Page 135]

Archiving and Deleting Idocs

Prerequisite

The IDocs to be archived have an archivable status [Page 125].

Procedure

1. Go to the central archiving transaction by choosing *Tools* → *Administration* → *Administration* → *Data archiving*. If necessary check the object specific Customizing [Ext.]:

   Here, for example, you can set the indicator "detailed statement" in the variant to the deletion program: This can be entered in the archiving log of each individual IDoc which is archived. However, this can lead to very large logs which do not necessarily contain useful information.

2. Enter the archiving object `IDOC` and choose the required action, here *Archive.*

   A warning will be displayed if some of the archiving sessions for the archiving object IDoc are still incomplete. You can ignore the warning but you then run the risk of possibly archiving some IDocs twice.

3. Define the *start date* of the archiving session and the *spool parameters* (print output).

   The traffic lights should now be green, indicating that these parameters have been maintained. Please note that you can set the archiving mode for the print output to *Store Only* and therefore avoid printing every individual archiving log. Enter the object type IDOC as the attachment parameter and use the F4 Help to select a suitable document type for the object type. Also enter a free text as *Additional information* for the archiving report.

4. Choose *Maintain* to create a report variant. As selection parameters, enter the message type and a time period within which the IDocs received their last, current status. Choose *Continue* and save your variant.

   You can select whether your IDocs receive a special status after archiving. No new status is set in the standard system.

5. Return to the archiving screen and choose .

   If you selected *immediately* in step 3, the message *New archiving job was generated* is displayed in the message bar at the bottom of the screen.

6. You can check whether your archiving session is complete by selecting *Job overview*. If you choose *Administration*, an overview of the completed and incomplete archiving runs is then displayed. If IDocs which match your selection parameters were found in the database tables, your archiving session is displayed as a folder full of archive files.

7. If not defined differently in object specific Customizing [Ext.], your IDocs are still located in the database after archiving, that is to say, your archiving session is **not yet complete**. To **delete** the IDocs, you must schedule a deletion job with the appropriate

Procedure

variant. Choose the archive file from which the IDocs are to be deleted from the database.

Reloading Data

Prerequisites

- You are currently in the *Archive Administration: Initial Screen* (transaction SARA). The name of the archiving object is in the *object name* field.
- Archive administration has access to the archive files you want to reload.

When reloading data, the system can only process complete archiving sessions.

Procedure

To reload data:

1. Choose *Goto* → *Reload*
   This takes you to the *Archive Administration: Reload Archive* screen, where you can schedule a background job.
2. Choose *Archive Selection*.
   A window containing the archiving sessions that have already been processed by the delete program appears.
3. Mark the desired archiving session and choose *Continue*.
   You return to the initial screen.
4. Maintain the start date and the spool parameters.
5. You have now entered all the data needed for the background job. Choose *Create Job* to create the job.

Reloading archived data to the database can cause problems, therefore you should only do this in an emergency. For example, if you establish immediately after archiving that you have archived the wrong data or too much data. You should, wherever possible reload this data immediately after archiving.

See also:

Archiving Data [Ext.]

Archive Selection [Ext.]

Displaying Archive File Details

Details [Ext.]

To display the status of a file:

1. Choose *Management*.
   The *Archive Administration: Archiving Session Overview* screen appears.
2. Position the cursor on the session.
3. Choose *Expand Subtree*.
4. Position the cursor on an archive file.
5. Choose *Choose*.

See also:

Archive Administration: Archiving Session Overview [Ext.]

Listing IDoc Numbers in Archive

1. Choose *Evaluate* in the initial screen of the central archiving transaction .
2. Choose *Execute.*

   The system displays the complete archiving sessions.
3. Select one or several sessions or files contained within and confirm your selection.

   A list of IDocs from the archive files selected is displayed. As well as the IDoc number, the "logical“ message, for example, is output (output fields of report RSXARCR [Page 137])

Single Display Using SAP AS (BC-SRV-EDI)

Use

You can display all archived IDocs by choosing the archive information system (SAP AS). There are two differences for IDoc search in the archive:

- The SAP AS displays all IDoc records, that is, the status records as well as the control and data records.
- You cannot use SAP AS to search for character strings in data records.

Prerequisites

The archive files are available in the system and are not stored using the Content Management Service (CMS).

The archive info structure is active and set up (see "activities")

Features

You use the archive info structure SAP_IDOC_001 in the standard system to search, for example, for

- Time (creation and last change)
- Partner
- Message
- IDoc number

You can also define your own structures. The field catalog containing the fields is called SAP_IDOC_001. It contains the fields from the IDoc control record EDI_DC, with the exception of client, ALE serialization and record type.

Authorizations

For single display, only the General Archiving Authorization [Ext.] is needed.

Activities

If necessary, you must activate the archive info structure [Ext.]. You can display the initial screen from the central archiving transaction [Page 126] by choosing *Information system* → *Customizing*.

If necessary, you must construct the archive info structure [Ext.]. You can display the initial screen from the central archiving transaction by choosing *Information system* → *Status*.

For single display you must evaluate the archive info structure [Ext.]. You can display the initial screen from the central archiving transaction by choosing *Information system* → *Archive Explorer*.

IDoc Search

Use

Users should be able to find IDocs not only via the address information or control information in the control record but also according to the business data they contain. You also want to be able to answer questions such as: “Which IDocs contain purchase orders for my material XY?”

The IDoc search function can be used to answer such questions. You can search for IDocs both in the database and in archive files.

Prerequisites

When searching in the archive (as opposed to searching in the database) you must

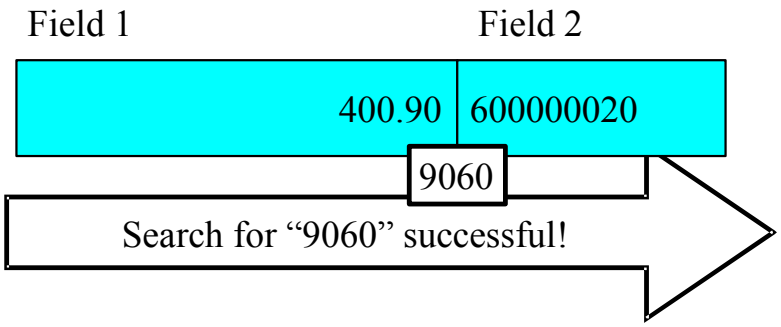
- Have deleted the IDocs from the database (complete archiving session)
- Have created an archive info structure in the central transaction SARA. For further information see Individual display via SAP AS (BC-SRV-EDI) [Page 132]

Features

The function searches for character strings, that is, you must enter the value of a segment field as it appears in the field itself (avoiding full stops and commas whenever possible).

You can enter a maximum of two values in two segment fields as selection criteria. If two values are entered, both must exist in the IDoc in the corresponding segment fields for the IDoc to be included in the list of results (normal AND link in selection fields).

You can also choose to only give the value instead of the segment field. This means that all data records (more precisely: the field SDATA in the data records) are searched for the character string, that is, the character string which is found may span several segment fields.



If you select *Quick search mode*, the search can be limited to a maximum of one hit in each IDoc, that is, the system terminates the search in the current IDoc after one hit.

The search can and should also be restricted by entering additional search criteria which require values from the control record (in the same way as the other monitoring programs): Creation time, partner and message, direction, and so on.

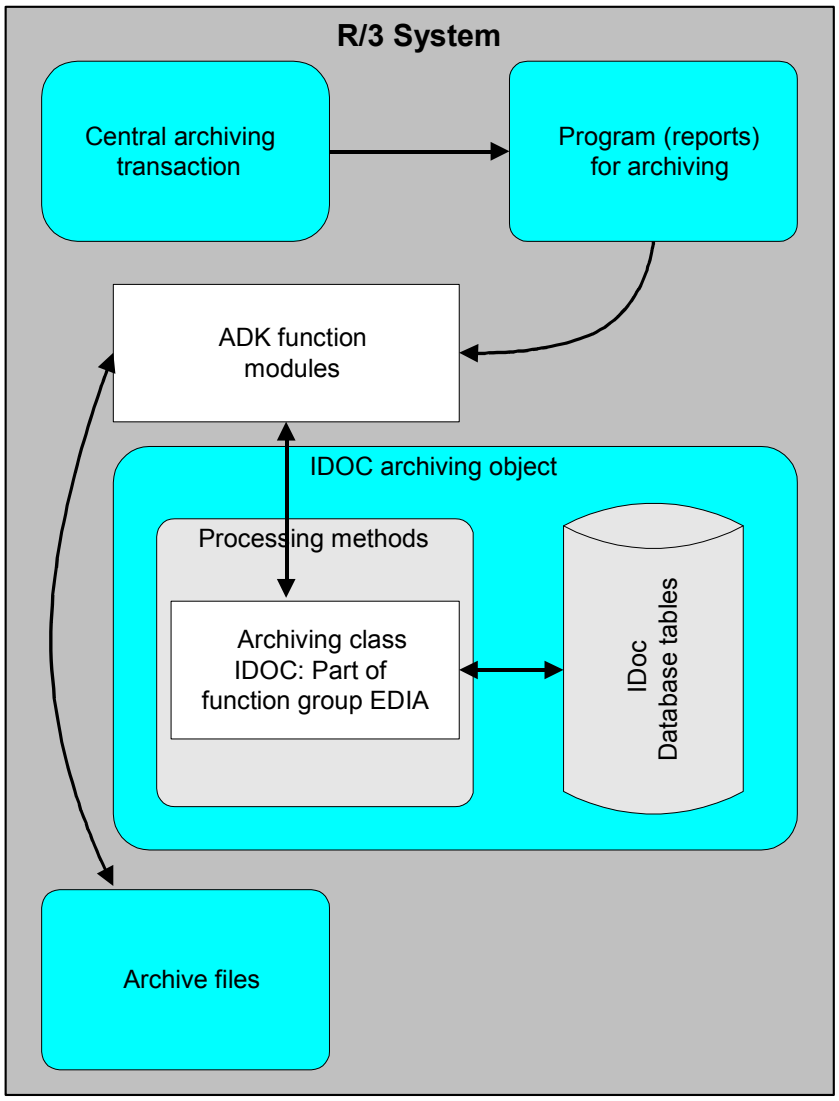
Activities

The IDoc Search can be accessed from the Initial node of the IDoc interface [Ext.] via *IDoc* → *Find IDoc*. Different transactions are called depending on whether you wish to search in the database or in an archive.

Archiving: Technical Background

The IDoc archiving tools were developed with the Archive Development Kit [Ext.] (**ADK**), which supports object-oriented programming methods. This has led to the use of **archiving object** IDOC, which contains the necessary **database tables** and the **archiving class** IDOC, which has the function modules required for processing.

The archiving object IDOC is accessed “from outside” via reports which (indirectly) call the ADK function modules. The selection parameters for these reports are configured in the variant maintenance function for the central archiving transaction. The ADK function modules, in turn, use the function modules from archiving class IDOC to access the database tables. The figure below illustrates this object-oriented structure.



All the function modules in the archiving class “IDOC” belong to the function group “EDIA”. This group also includes the function modules which the archiving programs use to call the ADK functions.

Archiving: Technical Background

In computer literature, data and the corresponding functions are sometimes combined in “classes” (for example, in C++). Here this would refer to the “archiving object” which contains the data (the lines in the database table), as well as the “archiving class” IDOC, which contains the corresponding functions.

The reports supplied with the standard system can be used as sample reports to write your own reports with selection parameters which meet your requirements. For more information on Standard Reports see

Archiving: Describing Standard Reports [Page 137]

Archiving: Describing Standard Reports

The following section contains information about the “initial reports” and the function module which they call from the function group EDIA. This function module, in turn, calls the ADK standard class function modules. The reports are addressed by the central archiving transaction SARA.

In order to address programs through the central archiving transaction SARA, you must replace the Standard Reports through the programs in transaction AOBJ.

Name	Description	Selection parameters
RSEXARCA	Writes IDocs to the archive. Calls EDI_ARCHIVE_IDOCS. Uses the subroutine INITIALIZE_STATUS_QUALITY (function group EDIA) to check whether the IDocs can be archived.	• Time at which the control record was last changed • Time of IDoc creation • Current status • Logical message • Direction • IDoc number • Partner
RSEXARCD	Deletes archived IDocs from the database. Calls EDI_DELETE_ARCHIVED_IDOCS.	Archive files from incomplete archiving sessions
RSEXARCR	Reads IDocs from an archive. Calls EDI_READ_IDOC _ARCHIVE. Displays the following control record fields for each IDoc: • Logical message • Status • Date and time at which the control record for the IDoc was last changed • Direction • IDoc number	Archive files from complete archiving sessions
RSEXARCL	Retrieves IDocs from the archive into the database. Calls EDI_RELOAD_IDOC _ARCHIVE. Checks for client or IDoc number conflicts via IS_RELOADING_POSSIBLE (subroutine for function group EDIA). Writes the new status “35” for outbound IDocs, “71” for inbound IDocs (“reloaded”). “Detailed statement” indicator as for RSEXARCA.	Archive files from complete archiving sessions
RSEXARCI	Creates index in table EDIDOCINDX. Calls ADK function module directly.	None: Reads the archive files of the IDOC object class.

RSEXARCJ	Removes index in table EDIDOCINDX. Calls ADK function module directly.	Date, up to which the indexes should be deleted. Number of records, after which a database commit should be transmitted.

Deleting Links with IDocs

Use

As the IDoc number increases, only the data records table gets bigger. The tables which contain links with application documents, for example, also get bigger and from time to time should be emptied of obsolete entries. Unlike the IDocs, links are not archived before they are deleted.

Procedure

1. Start the program RSRLDREL.
2. Enter a suitable *end date*.

   The end date is suitable if all links that were generated before are no longer needed. For example, if you have already archived and deleted [Page 127] all IDocs before this date and the application documents are also archived, you no longer need the corresponding links any more (exception: Application documents and IDocs are to be retrieved from the archive to the SAP System).
3. Select *Selection via link type* and enter a link type. Also refer to the table below.

   Alternatively, you can execute the selection by choosing *Object/Function*. If, for example, you want to delete all links with standard orders, then you should make your selections according to *object type* BUS2032 (standard order) in the *function* APPLOBJ.

If you want to trigger the following links...	then you must choose the link type...
Application document with outbound IDoc	IDC0
Inbound IDoc with application document	IDC1
Outbound IDoc with application document in the target system	IDC9
Inbound IDoc with application document from the front-end system	IDCB
Inbound IDoc with communication IDocs in distributed systems	IDC2
Copy of an IDoc with its original	IDC3 (inbound), IDC7(outbound)
Internet IDoc [Page 59] with folder from the Business Workplace	IDC5 (inbound), IDC6 (outbound)
Inbound IDoc with outbound IDoc from the front-end system	IDC4
IDoc with TID (port type tRFC [Page 37], ALE audit). Does not work with deletion criterion *Both objects not available*!	IDC8 (inbound), IDCA (outbound)
Outbound IDoc with reference IDoc (type IDCREF01)	IDRF

Deleting Links with IDocs

4. By using the *Deletion criteria* you can protect links from non-deleted objects from being deleted. The following scenarios are possible:

   - You want to delete links last. Choose *Both objects not available*. If now, for example, only the application document is available, you can actually display the numbers of the linked IDocs from the document, but no longer the IDocs themselves.

     This option does not work for links with TIDs (for ALE audit) although they appear to be there.

   - You want to delete the links if at least one of the two objects is no longer there (option *one object not available*). Then, for example, you can no longer display the numbers of IDocs that have already been deleted from an application document.

   - You generally want to delete all links that meet the other selection criteria. Select *Without Existence Check*.

5. Start the program.

   You can perform a *Test run*. Then the system only notifies you of the number of links that meet the selection criteria, but does not delete these.

Structure, Documentation and Definition of IDoc Types

Use

IDoc interface tools developed are used to define new IDoc types. The documentation tools are used to display the business structure of IDoc types as well as their technical structure in different formats. They help to determine whether new IDoc types really have to be defined or whether the types provided in the standard system are sufficient for your needs. They also help to automate the process of defining IDoc types in external systems.

Features

Structure of an IDoc [Page 142]

Documentation Tools [Page 146]

Defining New IDoc Types [Page 153]

IDoc Structure

Definition

All IDocs are created according to specific rules: This is the **general structure** (the record types) of every IDoc. Special rules affect the different **IDoc types**.

Structure

General Structure

IDocs contain **administration information** for technical processing, as well as the actual application data, which is stored in **segments**. A segment comprises **segment fields** as the smallest unit of the IDoc - comparable with the data elements from the EDIFACT standard.

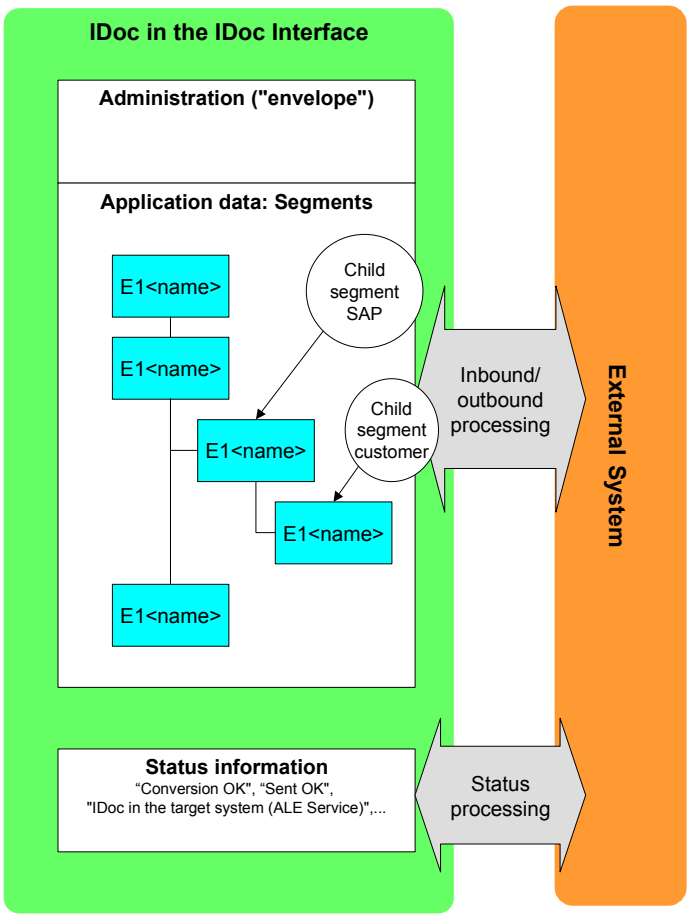
In the SAP system, the processing status (“what has happened to the IDoc before now?”) is stored in the IDoc **status information**. The status information also contains details about errors and indicates the data in which the error occurred. This status information is not forwarded as part of the IDoc but separately via “status processing”.

IDoc types (special structure)

An IDoc type is defined through its permitted segments. Segments can be dependent on each other (**parent and child segments**). For example, segment E1EDPT1 (document item text identification) is a child segment of segment E1EDP01 (document item data, general) in IDoc type EXPINV01 (export billing) and a child segment of E1EDP07 (order data shipping notification) in IDoc type DESADV01 (shipping notification). The segment, therefore, is used in several contexts and is the "child" of several "parents".

Customer extensions are created by defining new segments and appending them as children of existing segments of the basic type (the basic type is the IDoc type supplied with the standard SAP System, that is to say, the type to be extended). The new segments (and extension) must be within the customer namespace. SAP segment names start with “E1” (see Namespaces [Page 159])

The figure below shows the basic structure of an IDoc and which parts are transferred to and from the external system. The parent-child segment hierarchy is shown here as a tree structure. A total of 3 **hierarchy levels** can be seen.



The IDoc interface can check for each IDoc whether the segments contained are compatible with the definition of its type. This syntax check is activated or deactivated in the Partner profiles [Page 74] (separately for inbound and outbound processing.

See also:

IDoc Structure: Technical Implementation [Page 144]

Displaying the General Structure (Record Types) of an IDoc [Page 147]

Displaying IDoc Type or Segment Documentation [Page 148]

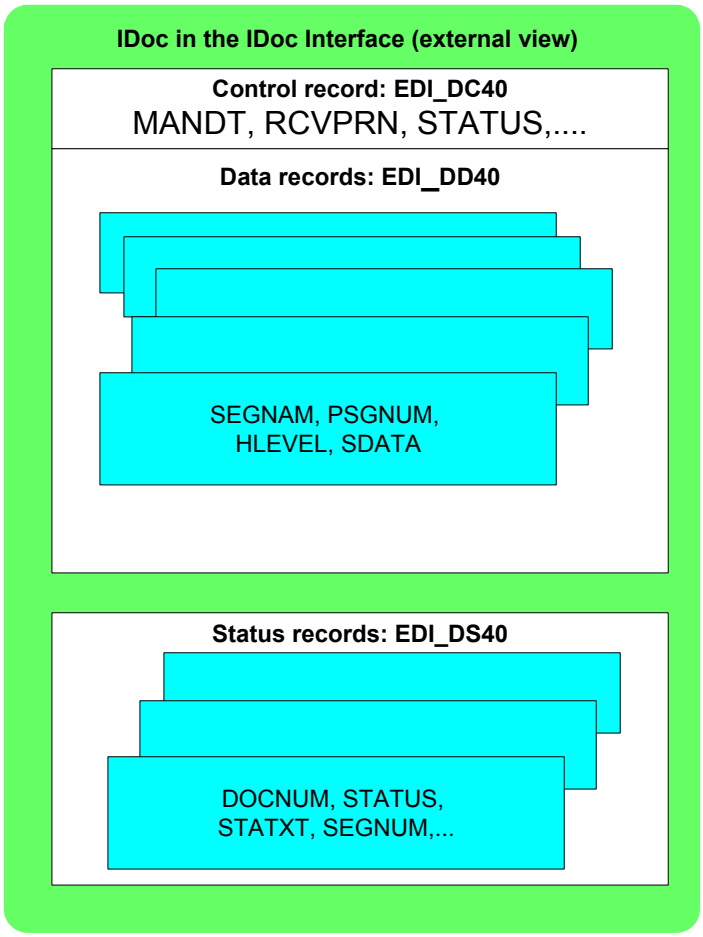
IDoc Structure: Technical Background

IDocs are archived in three database tables in SAP - one for the **control record**, one for the **data records**, and one for the **status records**.

- The control record is identical for all IDocs and contains the administration information, for example, sender, recipient and message. Note that the control record also contains the last processing status (STATUS field).
- The data records contain the segments and administration information for the segments (for example, a parent segment to which it is attached).
- The status records contain all of the previous processing statuses, as well as the corresponding administration information. This information is not sent between systems as part of the IDoc but rather as a separate data packet.

Viewed externally (that is, as they are sent to the subsystem), the IDocs are located in three structures which correspond to the three database tables: The control records, data records and status records are also used there. The description of the individual fields can be accessed in the system via the documentation tools: For further information, see Displaying the General Structure of an IDoc (Record Types) [Page 147].

The figure below shows the basic structure of the IDocs in the R/3 System (IDoc interface viewed externally, that is, the “external structures”) with some sample fields.



The parent-child structure of the segments is realized by the PSGNUM field, which contains the number of the parent segment (if applicable). In addition, the HLEVEL field also contains the hierarchy level. The actual application data, that is, the actual segment, is located in the SDATA field. This field, in turn, is subdivided into individual application fields. The maximum length of SDATA is 1000 bytes.

The SEGNUM field in the status record is used to identify segments which have caused errors. SEGFLD fulfills the same function at the lowest level of the segment fields.

A segment is defined as a required segment in the **IDoc type**, but was not filled when the actual **IDoc** was generated. For this reason, the relevant IDoc does not contain this required segment - the syntax check returns an error; the missing segment is marked in the SEGNUM field.

Documentation Tools

Use

The IDoc interface is open, which means that the structure and use of the IDoc types is transparent. To allow for this, the IDoc interface provides tools which display the documentation in various formats. You can choose between the **technical view** (such as field lengths or release information) and the **application view** (such as the meaning of the actual application data in the segment).

Another advantage of the documentation tools is that they can also format information in such a way that it is suitable for automatic processing. In this way, an EDI subsystem, for example, can be supplied with information about a new IDoc type so that the IDoc type can processed correctly in future.

Features

The following table provides an overview of what can be documented with each tool and in which format. You can call the tools from the Initial Node of the IDoc Interface [Ext.].

The *parser* format is a simple SAP list of fields to be documented which can be read by a parser in an external system. Associated units (segments, record types, etc.) are encapsulated by BEGIN...END commands. The list can be sent to a PC (formats: Rich text, ASCII, spreadsheet) transferred or printed.

The *tree display* format is used for output to the screen. The data can only be stored externally (for example, on a PC) in the other formats.

The format *DTD* is the *Document Type Declaration* for the Output of IDocs in XML files [Page 66]. It contains the XML elements which can or must appear in the IDocs of the type used.

Documentation of...	Supported formats	Access via...
General IDoc (record types)	HTML, C-header, IDoc, tree display (screen), parser	*Documentation → IDoc record types*
Specific IDoc type (for example ORDERS01)	HTML, C-header, IDoc, tree display (screen), DTD, parser	*Documentation → IDoc types* or 
Segment	HTML, tree display (screen)	*Documentation → IDoc segments*

Activities

Displaying the General Structure (Record Types) of an IDoc [Page 147]

Displaying IDoc Type Documentation [Page 148]

Documentation translation [Page 151]

Displaying the General Structure (IDoc Record Types)

1. From the Initial Node of the IDoc Interface [Ext.] choose *Documentation* → *IDoc record types* and select all three *record types*.
2. Choose the record type *version*.

   Release 4.0, for example, has version 3 (long names, customer prefixes: This is evident, for example, in the case of the field SEGNAM in the data record, which contains 30 characters in Release 4.0 and previously contained only 10).
3. Choose via *Goto* → *User settings* all *display attributes* for IDoc types. Save your settings and return to the initial screen.

   The screen is a tab page from IDoc administration [Page 247]. The *IDoc type attributes* are not selectable, as you are calling documentation that applies to all IDoc types!
4. Choose  for the tree display. This display option provides the clearest overview.

   The entire documentation for the structure of each IDoc, including the data elements which are used (in the following order: control record, data record and status record) is displayed. Note that typical IDocs contain several data records (segments) and status records.

Displaying IDoc Type or Segment Documentation

Prerequisites

- The following section assumes that you wish to display a specific **IDoc type**. To display the documentation for an individual **segment**, choose *Documentation* → *IDoc segments* from the initial screen of the IDoc interface and continue with step 3 in the section. However, the segment documentation only supports HTML and tree display formats.

  To display the documentation for a general IDoc type (**IDoc record types**), read the procedure in the following section - Displaying the General Structure (Record Types) of an IDoc [Page 147] You can display this documentation in all the formats specified below, apart from DTD.

- If you wish to send the documentation as an IDoc, you must define the IDoc type SYIDOC01 with the message type SYIDOC for your partner in outbound processing. For more information see: Partner Profiles in the Standard Dialog [Page 74]

Procedure

1. To gain an overview of all the IDoc types supplied by SAP (**basic types**), choose  in the Initial Node of the IDoc Interface [Ext.] and then the F4 Help for your object *basic type*.

   A list of the basic types supplied with your release version is displayed, together with a brief description.

2. Choose the required IDoc type.

3. Choose *Goto* → *User settings* and specify the required view. It means:

IDoc type attribute	Basic type or extension, release, version of record types: Also see important terms [Page 154]
Segment attributes	Frequency of use, necessity, qualified segment
Segment field attributes	Length (in bytes), data element, from which the field documentation is read.
Segment documentation	Defining text for the segment in general
Segment field documentation	Defining text for the individual segment fields = documentation of the associated data elements
Field values	For example fixed values which are attached to a domain or check tables from which the possible values are read.

   The further settings affect default values for different output formats (HTML, C-header and so on).

4. Return to the initial screen and choose  (display tree) for a quick overview of the system.

   The individual segments and their fields are displayed in a tree structure which corresponds to the parent-child segment hierarchy.

5. You can select within the tree structure which parameters are to be output for individual objects (for example, a particular segment field). You can switch the corresponding icon on or off by clicking on it. For an explanation of the icons, select  .
6. You can select additional formats from the initial screen.

    for the IDoc format. Specify the corresponding partner (partner number and partner type).

    for the HTML format. Three files are generated: One for the frame, one for the index and one for the actual documentation. Their names end correspondingly with _f, _i and _d. An HTML browser loads the frame file and places the index and documentation files in the corresponding parts of the frame. If you have made the settings in the IDoc Administration User Parameters [Page 247], the browser is started automatically.

    for the C-header

    for the parser

   Documentation → *Generate DTD* for the DTD format

   The C-header and HTML formats contain the information “released since release...”. In fact, this applies only from Release 4.0A onwards. In releases between 3.0D and 4.0A, this information specifies the release for which the IDoc type was **created** (which was usually where the IDoc type was also released). This information was not implemented in releases prior to 3.0D.

Displaying an IDoc Using an XSL Stylesheet

Use

You can represent IDocs in XML format individually (how you generate the Business Connector, for example) by means of a stylesheet. You can test the stylesheets by performing the steps specified below.

Prerequisites

You have put your stylesheet together with the graphics on the presentation server or in the Web Repository (transaction SMW0). In the Web Repository, graphics are identified as binary objects and stylesheets are identified as HTML templates.

Procedure

1. From the initial node of the IDoc Interface, choose *Development* → *IDoc styles*.
2. Identify your stylesheet and the graphics using a *Style ID*. See the F1 Help for the fields. Assign the style ID to the required message type.
3. Maintain an Inbound Partner Profile [Page 82] for the required message type. For the *process code* select **ED00_XML**. Assign yourself as the *permitted agent*.
4. Generate a test IDoc of the required message type, for example with the Test Tool [Page 93] and start processing.
5. You receive a work item in the Business Workplace - when you execute it you receive the XML display of the IDoc.

   You can select other stylesheets by using *View*.
6. Exit the work item by selecting *Edit*. Here you can also set the *Deletion indicator* with which the IDoc receives an Archivable (and thus deletable) Status [Page 125].

Documentation Translation

Use

For your own definitions (extensions, views or basic types) you can also enter documentation with the development tools of the IDoc interface (Segment- and IDoc type Editor). To translate this documentation into other languages, you must use the general translation transaction.

Prerequisites

- You have authorization for the translation transaction SE63 (authorization object S_TRANSLAT).
- You have your own extension, view or basic type which has been tested. For more information see

  Extending an IDoc Type [Page 172]

  IDoc views [Page 168]

  Defining and Using a New Basic Type [Page 189]
- You have a list which groups together the newly defined segments.
- You have a corresponding list of data elements which you use for your newly defined segment fields and which are not yet translated.
- For these data elements you have a list of stored domains, which have fixed values still to be translated.

Procedure

1. From the initial R/3 screen choose *Tools* → *ABAP Workbench*, *Utilities* → *Translation* → *Short and long texts*.

   You go to the general translation transaction.
2. *Only if you used data elements which have not yet been translated for your segment fields*: Choose *Translation* → *Short Texts* → *ABAP Dictionary* → *Data Elements.* Enter a data element, source and target language and choose *Environment with long texts.* Translate your data elements (short texts) and documentation (long texts). Save your text. Proceed in this way for all the data elements in your list.

   Segment fields are documented via data elements. Therefore you must translate both the short and the long texts in the translation transaction.
3. *Only if you used fixed values which have not yet been translated for your segment fields*: Choose *Translation* → *Short Texts* → *ABAP Dictionary* → *Values.* Enter a domain, source and target language and choose *Edit.* Translate your fixed values (short texts) and save the text. Proceed in this way for all the domains in your list.
4. Choose *Translation* → *Short texts* → *ABAP Dictionary* → *Table texts.* Enter a segment, source and target language and choose *Environment with long texts.* Translate the short description and the documentation (the long text) of your “table“. Save your text. Proceed in this way for all the segments in your list.

Segment types are stored in the dictionary as structures. As every table also has a structure, tables and structures are referred to as “tables“ in the translation transaction. Documented short and long texts are created for structures and tables.

You can also access translation in the Segment Editor [Page 193]: Here, choose *Environment* → *Translation*.

5. Choose *Translation* → *Logical objects* → *Logical objects*. Enter as object:

   - **IDOC**, if you have defined a basic type,
   - **IEXT**, if you have defined an extension
   - **IDCV**, if you have defined a view

   Enter a source and target language and choose *Edit.* Translate the short description and save the text.

   You can also access translation in the IDoc Type Editor [Page 167] or the IDoc Views [Page 168]: Here, choose *Environment* → *Translation*.

6. *Only if you have defined a new logical message:* Choose *Translation* → *Short Texts* → *Table Contents*. Enter **EDIMSGT** in the table and source and target language. Select *Edit.* Translate the description and save the text.

7. *Only if you have defined a new process code:* Choose *Translation* → *Short Texts* → *Table Entries*. Enter as table:

   - **EDE1T**, if you have defined a process code for outbound processing
   - **EDE2T**, if you have defined a process code for inbound processing

   Enter a source and target language and choose *Edit.* Translate the description and save the text.

Result

You have translated the documentation into your new definition. If you log on to the system in the target language, the documentation tools output the translated short and long texts. Your new logical messages and process codes are described in the target language in the IDoc interface.

Defining New IDoc Types

Purpose

If the IDoc types (**basic types**) supplied with the standard SAP System do not meet your requirements, you can define your own IDoc types.

Prerequisites

Before you define a new IDoc type, you should first check whether the basic types can be made to meet your requirements. You can display a list of these basic types and their meaning with the **documentation tools** of the IDoc interface, as described in point 1 of the section Display IDoc Type or Segment Documentation [Page 148]

Process flow

Preparation

New IDoc types should be defined within the scope of a **customer extension** within which customer segments are appended to an existing **basic type**. The section entitled

Important Terms [Page 154]

explains these and other important terms and should be read first.

Design Guidelines and Formatting Rules for Creating New Segments [Page 160]

When creating a customer extension, you should first list the segments which are required. You may have to develop your own segments, in which case you should follow these guidelines. The following section:

Design Guidelines for Creating New IDoc Types [Page 162]

should also be read before proceeding.

Definition

To define your IDoc types, you should use the **development tools** in the IDoc interface, which are described in the following sections:

Segment editor [Page 163]

IDoc Type Editor [Page 167]

The Programming Interface (API) for Defining IDoc Types [Page 169]

Automatic definition of IDoc types is only required in special cases. You can generally skip this section.

Check lists: From definition to implementation

The following two **checklists** describe a general procedure from definition to implementation for new IDoc types in two cases:

Extending an IDoc Type [Page 172]

Defining and Using a New Basic Type [Page 189]

Important Terms

Basic Type, Extension, IDoc Type

Definition

Some IDoc types are supplied by SAP in the standard system: These are the **basic types** Other IDoc types are customer extensions: In these cases, a basic type is combined with an **extension** which is created by the customer, according to certain rules. Unlike customer extensions (or modifications, see: Customer Extensions, SAP and Customer Developments [Page 157]), these extended basic types are upward compatible.

Different terms were used in 3.X Releases: The current basic type was previously called a “basic IDoc type” and an extension was called an “extension type”. The term “IDoc type” was used to denote both the extended basic type and the umbrella term.

Extensions are “appended” to basic types via **reference segments**. Since basic types cannot lose segments in subsequent SAP releases, compatibility is ensured.

Segment Type and Segment Definition

Definition

Segments form the basic building blocks of an IDoc type and are used to store the actual data.

A **segment type** is the name of a segment and is independent of the SAP release.

The **segment definition** is the release-specific name of a segment. By combining the segment type and the release, the required segment definition can be determined: This way, you can assign segment definitions from previous releases to an IDoc type in the current release. This may be necessary if, for example, the partner is using an older release which supports your current IDoc type but not your current segment definitions. You then have to “reset” these in the Partner profiles [Page 76] .

The segment definition is also the external name of the segment that the partner system "sees".

Do not transfer the meaning of these terms to IDoc types: Therefore, an IDoc type (“an IDoc is an IDoc type filled with data”) is always data-independent, while a segment type is always release-independent. A segment can be part of both an IDoc (data-dependent) and an IDoc type (data-independent).

Customer Extensions, SAP and Customer Developments

Definition

Customer extension - extended basic types are still supported after a release upgrade. Successors to basic types from previous SAP releases are automatically combined with the existing extensions. No manual maintenance is therefore required.

SAP development - new IDoc types created by SAP are supplied as new basic types. Manual maintenance on the part of the customer is not required after an upgrade.

Customer development (modification) - customer-defined basic types and any related extensions are not affected by an upgrade. These customer developments, therefore, may require subsequent maintenance.

Version Creation and Release Procedure

Definition

The following development objects in the IDoc interface can be release-specific, that is, they can exist in different **versions**:

- Basic types
- Extensions
- Segments (each **segment definition** is a version of the segment)

The basic type version is usually indicated by the last two digits. In the case of segments, the last three digits indicate the relevant segment definition.

Only one version of each object may exist per release. These versions have a unique **predecessor/successor relationship**.

An IDoc type, which is a combination of a basic type and an extension (an extended basic type) can therefore be version-specific in several ways:

- There are several versions of the basic type
- There are several versions of the extension
- There are several segment versions in the basic type
- There are several segment versions in the extension
- Any combination of the cases listed above

A successor must always have the same parts as the predecessor and at least one additional part: A new segment definition must therefore contain all the fields of its predecessor and at least one more. A successor of a basic type or extension must contain all segments of its predecessor and at least one more.

In order to be able to create a new successor, you must have **released** the predecessor. The object cannot be changed again after being released. The release can be canceled if no successors for the development object have been created.

Segment definitions which have not been released are assigned the default value “000” as a version number.

You should only release the objects when your development has been completed. You can also test objects which have not been released. You should only cancel releases in exceptional cases, since subsequent changes to objects which have already been used can lead to inconsistencies. Subsystems, in particular, will not support these changes.

Namespaces

Definition

Customer developments and SAP developments take place in different **namespaces**, which are defined by the permitted initial letters of the respective development objects. From Release 4.0 onwards, customer prefixes and SAP prefixes, for example, “/sap/” or “/<customer>/”, are also supported.

Use

The following table contains a list of the namespaces for the relevant development objects in the IDoc interface.

Object	SAP names start with...	Customer names start with...
Basic type	A-X or SAP prefix	Y-Z or prefix
Extension	Not supplied by SAP	No restrictions
Segment	E1	Z1 or prefix
Message type	A-X	Y-Z or prefix

Design Guidelines and Formatting Rules for Creating New Segments

Design guidelines

- Do not re-invent the wheel

  Before you create a new segment, check whether your requirements can be met by an existing segment. Use the EDI standards, for example, ANSI X12 (North America) or EDIFACT (worldwide). You should use ISO codes for coded field values so that they are based on a standard which is generally applicable. ISO codes exist, for example, for countries (ISO 3166), currencies (ISO 4217) and units of measure (ISO 31).

- Group your data in a meaningful manner

  Business data is contained in segment fields. Data which belongs together from a logical point of view should be combined to form a segment. Segments which belong together from a logical point of view should be combined to form a segment group. For example, the materials in a purchase order could be contained in one segment. Together with other segments (information about the partner, taxes and terms of payment), this segment can then form part of the “Items” segment group.

- Create universal segments

  You can use segments which you have already created in other IDoc types. Check whether a segment can be used for different messages. An example of a universal segment is one which contains general address data.

- Create clearly arranged segments and segment groups

  Position the required fields as close as possible to the start of the segment. Two segments which always occur together should be combined in one segment.

  Consider whether the use of qualifiers can be avoided. **Qualifiers** indicate how a segment is used and are difficult to document and interpret. Qualifiers are not usually defined on a cross-application basis, with the result that these segments are not universal.

- Don’t waste space

  The maximum length of all the fields in a segment is 1000 characters. The larger the segment, the better the ratio of useful data to administration data. A certain amount of space should be reserved in segments to allow for additional fields in the future.

- Document your segments

  Use the documentation options provided by the segment editor. You can store an additional data element for each field to provide information about how the field is used in the application. You can also document a segment to explain the attributes and structure of the entire segment. When documenting segments, remember that the segment fields are used in both inbound and outbound processing.

- Remember that conversion to EDI standards may be necessary

  Note the following formatting rules which specifically apply here.

Formatting rules

The following formatting rules usually apply to data in segment fields:

- The segment fields may only contain alphanumeric data. You can make sure that this is the case by using data elements which contain characters in the segment fields.
- No distinction is made between upper case and lower case letters.
- The fields are maintained in such a way that the values are left-justified.
- The standard date representation (without qualifiers) is YYYYMMDD. February 28, 1996 is therefore represented as *19960228*.
- The standard time representation (without qualifiers) is HHMMSS. *8.35pm* is therefore represented as *203500*.
- Values with a fixed decimal point are represented with a decimal point only - there is no thousand delimiter. Negative values are indicated by a minus sign after the value.

  German use: The number -2,097,152.64 is represented as *2097152.64-.*

  American use: The number -2,097,152.64 is represented as *2097152.64-.*
- Floating point numbers are represented with a decimal point - there is no thousand delimiter. Negative values are indicated via a minus sign after the value and the exponent is always specified.

  German use: The number -23.4 is represented as *-2.34E+01.*

  American use: The number -23.4 is represented as *-2.34E+01.*

The following rules may be useful if your IDocs are to be converted to other EDI standards (EDIFACT, ANSI X12):

- If possible, segment fields in the IDoc should be represented directly by EDI data elements. Avoid splitting or combining fields or other changes which could lead to problems with automatic conversion.
- The field length should be equal to the maximum length of the SAP application or relevant EDI standard (depending on which is longer), so that all the existing information can always be transferred.
- The required segment fields in the EDI standard must be present in the IDoc segment, to ensure that the IDocs can be converted correctly.

Design Guidelines for Creating New IDoc Types

- Create clearly arranged IDoc types

  Your IDoc type should have a flat structure (not too many parent-child generations). A segment may be used only once to define an IDoc type. However, since segments or segment groups can be repeated (minimum and maximum frequency of use are two of their attributes) and can be ‘optional’ only (attribute: optional segment), they may appear several times in the generated IDoc or not at all. Position the required segments as close as possible to the start of the IDoc type.

- Remember that conversion to EDI standards may be necessary

  The required segments in the EDI standard should therefore correspond to those in the IDoc type. Similarly, the minimum and maximum frequency of use should be selected accordingly.

- Don’t forget the assigned function modules

  The posting module must support all the permitted IDocs in the relevant IDoc type. Similarly, the selection module must be able to create the segments in the correct sequence and hierarchy. Otherwise, the syntax check of the IDoc interface detects an error.

Segment Editor

Use

The segment editor is used to create and change segments. Segments are always changed at Segment definition [Page 156] level, that is to say, the version of the segment used in the current release. The segment editor ensures that the rules for creating segment definitions are observed, for example, that only one segment definition may exist per release.

Features

The segment definitions are grouped together from the individual fields and the associated data elements and domains. In addition, **application documentation** can (and should) also be created for the segment definitions. An overview of the existing segment definitions is displayed immediately. The **where-used list** contains all the IDoc types in which the relevant segment is already used.

Once development of a segment definition has been completed, the definition is **activated** and **released** by the editor. When the definition is activated, all of the associated structures in the ABAP Dictionary are also activated at the same time. The release of the segment definition is a safety concept: Once released, further changes cannot be made to the segment definition.

The segment editor includes an automatic **transport link**. When you define or change objects, you are automatically asked to create a transport request. You can avoid this by configuring a **standard request**, to which all changes made in any one session are assigned. The standard request can be configured directly from the segment editor.

Activities

Changing Segments [Page 164]

Defining segments [Page 193]

Communication with Older Releases [Page 33]

Changing Segments

Prerequisites

Segments are changed via their **segment definitions**. Only one segment definition may exist for each SAP release. If a segment already has a segment definition from the current release, the segment can only be changed by overwriting the current segment definition.

Procedure

1. From the Initial Node of the IDoc Interface [Ext.], call the segment editor by choosing *Development → IDoc segments*. Enter the segment type and choose ✔.

   The editor displays the existing segment definitions with the **release indicator** and the relevant release.

2. Before you change the segment, you should create the **transport request** for your new segment definition and the relevant documentation. You can access the Workbench Organizer from the segment editor by selecting *Utilities → Requests (Organizer).* From there, select and reply to the questions which appear on screen. By selecting you can return to the segment editor.

3. If the last segment definition is from your release and is also released, you must reset the release indicator by selecting *Edit → Cancel release*.

4. Edit the last segment definition which has not been released via or create a new segment definition for the current release via .

   If the edited segment definition has a **predecessor** (that is to say an earlier version), the fields from the predecessor are displayed in gray in the editor and cannot be changed. You can only attach new fields to the segment (predecessor-successor relationship).

5. The subsequent steps (edit, save, release, transport) are identical to those described in the section: Defining Segments [Page 193] . The version of your new segment definition is 1 higher than the last segment definition.

Defining Segments

This section involves a scenario in which you have to create a new segment. If you have to create a new segment definition, change an existing segment [Page 164] using the segment editor.

Prerequisites

You must have authorization to create development objects and transport requests.

Procedure

1. Call the segment editor from the Initial Node of the IDoc Interface [Ext.] by selecting *Development → IDoc segments.* Select *Utilities → Requests (organizer)* to create a request.
2. Specify a segment type [Page 156] and select  .
3. Specify a description of the segment and enter the field names and data elements: Decide whether your field is to store data which is coded according to the ISO standard (checkbox).

   The field lengths (in bytes) and any fixed values are derived from the data elements via the domains (with field lengths as *export lengths*). The documentation for the data element is also copied and displayed by the documentation tools.

   Do not change the export length. Data is sent to and from the external systems as *character* fields with this length. However, internally (in the IDoc), the data is stored as a field of the same type as the selected data element (for example as an *integer*) with the corresponding length. Different lengths can lead to a loss of data (superfluous bytes are cut off).
4. Choose *Goto → Documentation* and describe the **segment type**. Save the active version of your **application documentation** and return to the segment editor.

   If you use the segment later in an IDoc type and display this type via the documentation tools, the description you have entered here will be displayed.
5. Save the segment definition [Page 156] by selecting  . Enter the segment in a transport request.

   You should assign all your development objects (segments, IDoc type, function group, and so on) for IDoc types to the same development class, so that the transport properties are always identical.
6. Exit the screen (F3).

   The segment definition you have created appears with the version number 000 in the segment editor. It is also included in the transport request you created.

Result

You have now created a new segment type with one segment definition. Although not released [Page 158], you can still test and work with the segment type. **You should only release the segment after you have successfully tested the processing.**

In the example, you should define two segments - E1HEAD (header) and E1ITEM (item). These two segments already exist in the system. They are only used as examples and should not be used in live systems. For example, the segment E1HEAD starts with the following fields:

Field	Data element	Length
BELNR (document no.)	EDI_BELNR	35
VKORG (sales org.)	VKORG	4
VTWEG (distrib. channel)	VTWEG	2
SPART (division)	SPART	2
...	...	...

IDoc Type Editor

Use

You can edit both basic types (SAP or customer development) and extensions (customer extension).

Features

Apart from two header entries (name and description), IDoc types are edited at the **segment** level. Segments can be appended, inserted or deleted again. The IDoc type editor also allows you to call the segment editor.

The IDoc type is represented as a **tree** structure in which the parent-child structure of segments is clearly displayed (see also: General Structure of an IDoc [Page 142]): A parent segment is displayed as a **node** which contains branches to the individual children.

Once you have completed development of an IDoc type, you realease it. The **release,** as in the case of the segment definition, is a safety concept: Once released, further changes cannot be made to the IDoc type. Before the IDoc type is released, however, you should use the editor to check whether any inconsistencies exist at the segment level - this prevents incorrect segments from being inserted, for example.

You can assign a **message** type to the new IDoc type directly from the IDoc type editor. You must do this anyway, as the system checks this assignment later in the partner profiles.

The IDoc type editor includes an automatic **transport link**. When you define or change an object, you are automatically asked to create a transport request. You can avoid this by configuring a **standard request**, to which all changes made in any one session are assigned. The standard request can also be configured directly from the IDoc type editor.

Activities

The IDoc type editor can be used to extend an existing IDoc type or to create a new IDoc type. In order to be able to actually use a new IDoc type (extended or new basic type), however, you must carry out further steps Refer to the following sections:

Two checklists:

Extending an IDoc Type [Page 172]

Defining and Using a New Basic Type [Page 189]

IDoc Views

Use

IDoc types can be used for more than one message, that is, for more than one business process. As a result, the IDoc types usually contain more segments than necessary for the individual application cases. To improve performance when generating IDocs, you can use views to ensure that only the segments relevant to the current business process are filled with data. Views are therefore only important for **IDoc outbound processing**.

Integration

The application must support this function: The program which writes the application data in the IDocs must carry out the following tasks:

- Determine from the partner profiles whether a view exists. To determine this, the function module EDI_PARTNER_READ_OUTGOING is called.
- Query, which segments belong to the view. The function module EDI_VIEW_READ is called once, which returns the segments in a table.
- Query, whether the current segment should be maintained in the current view. The table returned by EDI_VIEW_READ is used.

The standard delivery system contains views for the IDoc types DELVRY01 and INVOIC01.

Prerequisites

You must have IDoc development authorization (authorization object S_IDOCDEFT, for example in the role SAP_BC_SRV_EDI_DEVELOPER_AG) before you can define a view.

Activities

1. In the Initial Node of the IDoc Interface [Ext.] choose *Development* → *IDoc view* to define a view. Enter a name for the view and select  .
2. When the next screen appears, assign a message type (logical message) and a basic type to the view (the assignment of an extension to the view is optional). This assignment is checked in the partner profiles.
3. Position the cursor on a segment, which is to be included in the view. Choose  . The following segments must be included in the view:
   a) Qualified segments
   b) Mandatory segments
4. Save your entries.
5. In the partner profiles (General outbound processing [Page 76]), enter the view for the corresponding combination of partner and message.

API for IDoc Types and Segment Definitions

Definition

A wide range of function modules is available for developing segments and IDoc types automatically. This programming interface (API) is also used by the IDoc type editor and the segment editor.

Use

IDoc types are used in BAPIs [Ext.] to transport business information. When a new BAPI is developed, the corresponding IDocs are also generated via this API.

Structure

The API is divided into three function groups: The function modules which operate on segments are located in function group EDIJ, while those which operate on basic types and extensions are located in function group EDIM. The third function group contains RFC-compatible function modules which can be used from external developments. One example is the IDoc Class Library [Ext.] (the documentation is only available in English), which is written in C.

Function modules in the EDIJ group (segments)

Name	Description
SEGMENT_CREATE	Creates a new segment. The segment structure is transferred as a table. The new (and up to now only) segment definition has the temporary release indicator <SPACE>. The first version (Segment definition [Page 156]) is 000.
SEGMENT_MODIFY	Changes an existing segment. Depending on the **release indicator**, a new segment definition is created or an existing one modified.
SEGMENT_DELETE	Deletes a complete segment.
SEGMENTDEFINITION_DELETE	Deletes a segment definition which has not been released.
SEGMENTDEFINITION_CLOSE	Releases a segment definition, that is, sets the release indicator. The released segment definition is assigned the identifier from the current release.
SEGMENTDEFINITION_UNCLOSE	Cancels the release, that is, deletes the release indicator. The relevant segment definition is assigned the release indicator <SPACE>.
SEGMENT_CHECK_EXISTENCE	Returns a table with the existing segment definitions with the release indicators.

API for IDoc Types and Segment Definitions

SEGMENT_CHECK_CONSISTENCY	Among other things, checks whether the individual segment fields can be added seamlessly to the segment, whether the segment is included in a transport request, whether the last user to change a field in the dictionary is identical to the last user to change this field in the segment.
SEGMENT_TRANSPORT_INSERT	Inserts a segment with all the relevant segment definitions in a transport request. CAUTION: A request and task number must be entered.

Function modules in the EDIM group (basic types and extensions)

<OBJECT> represents the object on which the function module operates: IDOCTYPE for basic type, EXTTYPE for extension. Since an extended basic type consists of a basic type and an extension, separate function modules are not required.

Name	Description
<OBJECT>_CREATE	Generates an existing basic type or extension using a syntax description provided in a table. Also checks whether the object already exists. The new object is then included in a transport request.
<OBJECT>_UPDATE	Changes an existing basic type or extension using a syntax description provided in a table.
<OBJECT>_READ	Reads all the information for an object: Syntax, any extensions or predecessors.
<OBJECT>_DELETE	Deletes a basic type or extension.
<OBJECT>_CLOSE	Releases an object, that is, sets the release indicator.
<OBJECT>_UNCLOSE	Cancels the release of an object.
<OBJECT>_EXISTENCE_CHECK	Checks whether the transferred basic type or extension already exists. If so, the attributes are read (links to basic types, predecessors, and so on).
<OBJECT>_INTEGRITY_CHECK	Checks the general syntax. In the case of successors, checks whether they have been created correctly from predecessors.
<OBJECT>_TRANSPORT	Includes the basic type or extension in a transport request.

Function modules in the EDIMEXT group (API for external programs)

Name	Description
IDOCTYPES_FOR_MESTYPE_READ	Reads all IDoc types assigned to a message type (logical message).

IDOCTYPE_READ_COMPLETE	Reads the structure and attributes (segments), as well as the segment attributes (fields and fixed values), for an IDoc type. In this case, the version of the record types [Page 147] and segments [Page 158] must be sent to the function module.
IDOC_RECORD_READ	Reads the structure of the record types for the specified version.
SEGMENT_READ_COMPLETE	Reads the structure and attributes for a segment. In this case, the version of the record types and the release for the segments must be sent to the function module.

Extending an IDoc Type

Purpose

In the case of a customer extension, you add additional segments to an IDoc type supplied by SAP (basic type), so that you can transfer more (or simply other) business information. This extension is upward compatible, that is, it will be supported by future releases.

Prerequisites

This checklist requires that the basic type to be extended is already included in the processing: This means, in particular, that the processing (inbound and outbound) is defined via process codes in the partner profiles and that the corresponding programs (for example, function modules) which lie behind the process codes are extended in the same way as the basic type.

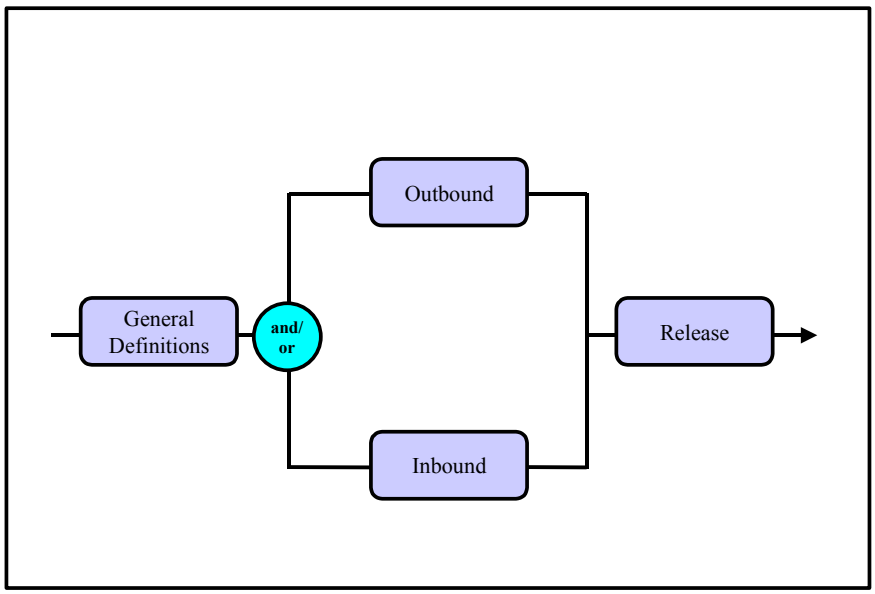
You require development authorizations (authorization object S_IDOCDEFT, for example, in the role SAP_BC_SRV_EDI_DEVELOPER_AG) for the extensions. Development takes place in the customer system.

Process flow

The following subsections contain the individual steps. The list can be used as a checklist when creating the extension. More information about the required steps and any alternatives can be found in the process flow diagram.

With the help of an **example**, you can define an actual IDoc type and process the type in a number of different ways. The example is used in each individual step. As a prerequisite, master data for the test customer TESTCUST (partner type KU) and the test vendor TESTVEND (partner type LI) must be present in the system. In addition, the example also assumes that the file port TESTPORT has been created.

Process flow diagram



Individual steps

General Definitions

Combining Segments [Page 174]

Extending Basic Types [Page 175]

Assign Message Type Basic Type and Extension [Page 177]

Configuring outbound processing (outbound under Message Control)

Extending Outbound Function Modules [Page 178]

Changing Partner Profiles (Outbound) [Page 180]

Testing Outbound Processing [Page 207]

Configuring inbound processing (direct inbound processing via ALE function module)

Extending Inbound Function Modules [Page 182]

Extending Function Module Assignments (Direct Inbound) [Page 184]

Checking Partner Profiles (Inbound) [Page 185]

Testing Inbound Processing [Page 242]

Releasing New Objects [Page 245]

Combining Segments

Prerequisites

You require development authorizations (authorization object S_IDOCDEFT, for example, in the role SAP_BC_SRV_EDI_DEVELOPER_AG) for new definitions. Development takes place in the customer system.

Procedure

You can:

- Copy SAP segments which you wish to use in your extension into your namespace [Page 159] (= “Create with reference”).
- Create segments without reference.

The following section assumes that you are creating segments without reference, that is, defining new field combinations.

1. Call the segment editor from the Initial Node of the IDoc Interface [Ext.] by selecting *Development → IDoc segments.* Select *Utilities → Requests (Organizer)* to create a request. You will assign all development objects to this request later. Return to the segment editor.
2. Name your segment type [Page 156] and select  . See also: Namespace [Page 159]
3. Describe your segment and enter the field names and data elements from which the field lengths (in bytes) and any fixed values are to be determined via the domains. The documentation for the data element is also copied and displayed by the documentation tools.

   You should still enter a specific documentation for your segment type via *Goto → Documentation*.

4. Save the segment definition [Page 156] by selecting  . Enter the segment in a transport request.
5. Exit the screen (). The segment definition you have created appears with the version number 000 in the segment editor.

   In the example, your segment has the name Z1TEST1 and has the following fields:

Field	Data element	Length
Continent	CHAR20	20
Country	CHAR2	2

Extending a Basic Type

Prerequisites

You must have completed the required steps in Extending an IDoc Type [Page 172] .

It can occur that you want to fill table fields which already have an equivalent in the basic type but which are not supplied in the standard system. You therefore do not need to extend the basic type, but rather only the function module that supplies the application fields. In this case, skip the procedure.

Procedure

1. From the initial node of the IDoc interface, call the IDoc type editor by selecting *Development → IDoc types*.
2. If you have not already done so, create a **transport request** for your development objects (extension segments, extension, function exit and so on). You can access the Workbench Organizer from the editor by selecting *Requests (Organizer).* From there, select *Request → Create* and reply to the questions which appear on screen. By selecting  you can return to the IDoc type editor.
3. Name your extension in accordance with the customer namespace and choose *Extension* as the development object. Choose  .

   The name of your extension should be no more than 8 characters if you wish to use the extension to communicate with partners who are using SAP releases prior to 4.0. You can only use extension names with more than 8 characters from Release 4.0 onwards.
4. Choose one of the following options:
   - *Create New:* Create the extension without a reference.
   - *Create copy,* which you can then change if necessary
   - *Create as successor*: Note that only one successor is allowed per release for each extension.
5. Select the basic type which is to be combined with your extension. Confirm your entries.

   The basic type is displayed as a tree structure. See the color legend (*Utilities → Color legend*).
6. Position the cursor on the segment where you wish to append the extension segments (the **reference segment**). Choose  . You must now assign **attributes** for these child segments:
   - Mandatory segment (indicator): Must data be written to this segment in the IDoc?
   - Maximum and minimum frequency of use: how often may / must a segment occur in the IDoc?

Extending a Basic Type

7. You can add further children or “grandchildren” to this or any other reference segment in the basic type.
8. When the new extension is complete, you can save your entry.

You should bear the following factors in mind when dealing with extensions:

- *Segments must not explicitly occur more than once in the IDoc type* - this means that the extension cannot have any segments which already exist in the basic type. This also means that a successor must not have any segments which already exist in the predecessor.
- Parent segments of predecessors must remain parent segments in the case of successors.
- Extension segments cannot be created from reference segments using the *Cut* and *Paste* functions and vice versa.

In the example call your extension Z1ORDERS. Append child segment Z1TEST1 to reference segment E1EDP19 in basic type ORDERS01. The child segment can be used only once.

Assign Message Type Basic Type and Extension

1. From the Initial Node of the IDoc Interface [Ext.] choose *Development* → *IDoc Type/Message*.
2. Select and then *New Entries*.
3. Enter the basic type, its extension and the message type to which you are assigning this combination. Specify the current release.

   The partner profiles check whether the IDoc type and message type “match“ each other. You must therefore make this entry. You also get a specific F4 Help from the mapping: All assigned IDoc types are offered in the corresponding field for each message type and vice versa.
4. Save your entries.

In the example, you should maintain the following mapping:

Message type	ORDERS (supplied with the standard system)
Basic type	ORDERS01
Extension	Z1ORDERS
Release	45A

Extending Outbound Function Modules

For each extension segment, there is a customer extension in the coding of the outbound function module, which enters the application data in the extension segment.

Prerequisites

You must have completed the required steps in Extending an IDoc Type [Page 172] .

Procedure

1. You write the customer extension of the corresponding function modules via a function or user exit in the project management: From the initial screen of the ABAP Wokbench choose *Utilities* → *Extensions* → *Project management*. Enter a project name (note the customer namespace!) and choose *Attributes* (radio button) and *create.* Enter a short text and save it.

   More detailed information about enhancement projects can be found in the R/3 Library under Basis → ABAP Workbench → Changing the SAP Standard → The R/3 Enhancement Concept [Ext.].

2. Return to the initial screen of the project management and select the sub-object *extensions allocation*. Choose *Change* to assign SAP extensions to your project. Use the F4 Help to find the SAP extensions. Save your entries.

   An SAP extension contains as components function exits (function modules), from which you will choose those required in the next step.

3. Now choose the sub-object *Components* in the initial screen of project management. Choose *Change.*

   A list of all the function exits is output. By selecting *Extension,* documentation on the individual Exits is displayed.

4. By double-clicking on the function exit required, the ABAP Editor is displayed.

5. The function exit contains the line **include <customer include>**. Create this *include* by double-clicking on the name (confirm system queries). Enter your source text and then save your entries.

   You do not have to maintain the fields HLEVEL (hierarchy level of the segment) and PSGNUM (number of the parent segment) in the IDoc data record. This is done by the IDoc interface, which can obtain the information about the position of the segment in the IDoc type from the definition of the extension.

6. You must now define your new extension segments as global structures in the customer-specific TOP include. Choose *Goto* → *Global data* in the ABAP Editor.

   The TOP program is displayed.

7. Create the customer TOP Include by double-clicking on the name (confirm system queries!). Enter your extension segments as TABLES data declarations.

If you have activated the segment before, there should be no problems with the syntax check ( ). You should never create the segment directly in the Dictionary! **Always** use the segment editor!

8. Activate your entire project. To do this access the initial project management screen and select .

In the example, extend the function module IDOC_OUTPUT_ORDERS in the additional project ZM06E001.

SAP extension	MM06E001
Function exit	EXIT_SAPLEINM_002 (Customer extension of data segments for outbound purchasing document)
Include	ZXM06U02
Segment	Z1TEST1
Main program	SAPLXM06
TOP program	LXM06TOP
Customer Top Include	ZXM06TOP

The source text appears as follows:

```
DATA segnam(27).
segnam = int_edidd-segnam.
CASE segnam.
   WHEN 'E1EDP19'.
   CLEAR int_edidd.                  "Clear work area
   int_edidd-segnam = 'Z1TEST1'.     "Name of customer segment
   z1test1-continent = 'Europe'.
   int_edidd-sdata = z1test1.     "Remaining fields from int_edidd
      "can be copied
   APPEND int_edidd.
ENDCASE.
```

The **CASE** instruction in the coding is essential because the same customer exit is called for each segment in the function module IDOC_OUTPUT_ORDERS. For this reason, the new *include* must first determine whether the call came from the correct parent segment.

Changing Partner Profiles (Outbound)

The extension must be entered in the outbound partner profiles before being used.

Prerequisites

You must have completed the required steps in Extending an IDoc Type [Page 172] .

Procedure

Extend your existing partner profiles. For more information about the procedure, see Partner Profiles in the Standard Dialog [Page 74]

In the example, you should maintain the following fields in the partner profiles for “General outbound” and “Outbound processing using Message Control”:

Field	Value
Partner number	**TESTVEND**
Partner type	**LI**
Partner function	**VD**
Message type	**ORDERS**
Basic type	**ORDERS01**
Extension	**Z1ORDERS**
Receiver port	**TESTPORT**
Output mode	*Transfer IDocs immediately*, *do not start subsystem*
Permitted agent	Your SAP user, type **US** (individual user), language **EN**
Application	**EF** (purchasing)
Output type	**NEU (new)**
Process code	**ME10**

Testing Outbound Processing

Prerequisites

You must have completed the required steps in:

- Extending an IDoc Type [Page 172] (if you have extended a basic type) **or**
- Defining and Using a New Basic Type [Page 189] (if you have defined a new basic type)

Procedure

The IDoc is generated from the application. If you have configured the Dispatch Time [Ext.] as “1” in the condition record for outbound processing under Message Control (MC), only the MC record is generated. You can test the MC record using the program Test from MC [Page 97]

In the example, create a purchase order (transaction ME21). The recipient is TESTVEND, you are ordering a material from the material master. In the condition record (processing in the SAP purchasing transaction: *Header* → *Messages*) you have selected dispatch time 4 (processing via *additional specifications)*, that is, the IDoc is generated immediately. In segment Z1TEST1 it shows the value “Europe“ in field CONTINENT: You can check this using IDoc Display [Page 117].

Extending Inbound Function Modules

A customer extension in the coding of the inbound function module which maintains the application tables with data from the extension segments relates to the extension segments.

Prerequisites

You must have completed the required steps in Extending an IDoc Type [Page 172] .

Procedure

The customer extension to the corresponding function modules is written via transaction CMOD. You proceed in a similar way as to Outbound processing [Page 178].

In the example, extend the function module IDOC_INPUT_ORDERS in the additional project ZEDA0001. The extension writes the value from the segment field Z1TEST1-KONTINENT in the field IHREZ (customer character) of structure XVBAK. The contents of this structure is used in the structure of batch input table BDCDATA, which transfers transaction VA01 (create standard order).

IHREZ is already supplied from field E1EDKA1-IHREZ of the basic type ORDERS01, therefore an extension here is not necessary at all. The example should just be used as an illustration of the principle.

SAP extension	VEDA0001
Function exit	EXIT_SAPLVEDA_001 (Customer extension of data segments for inbound order document)
Include	ZXVEDU03
Segment	Z1TEST1
Main program	SAPLXVED
TOP program	LXVEDTOP
Customer Top Include	ZXVEDTOP

The source text appears as follows:

```
DATA segnam(27).

DATA: BEGIN OF hxvbak.

   INCLUDE STRUCTURE vbak.

DATA END OF hxvbak.

segnam = segment-segnam.

CASE segnam.

      WHEN 'Z1TEST1'.                                    "Name of
      customer segment
```

```
      z1test1 = segment-sdata.                          "Structure for
      data

      hxvbak-ihrez = z1test1-continent.   "Fill help structure

      MOVE hxvbak TO dxvbak.                            "supply CHANGING-
      Parameter

ENDCASE.
```

The **CASE** instruction in the coding is essential because the same customer exit is called for each segment in the function module IDOC_OUTPUT_ORDERS. For this reason, the new include must first determine whether the call came from the correct parent segment.

Extending Function Module Assignments (Direct Inbound)

Here you specify that your extended function module may process your extended basic type with a specific message. This involves enhancing the **ALE registration** of the function module.

Prerequisites

You must have completed the required steps for outbound processing in Extending an IDoc Type [Page 172] .

Procedure

1. From the Initial Node of the IDoc Interface [Ext.], choose *Development* → *Message/appl. object,* , *New entries*. Assign the relevant function module (type F) to the basic type, to your new extension and to your message type and select the application object as in a non-extended case.
2. Enter *direction* `2` (inbound). Save your entries.

In the example, the following objects are assigned to each other when you have made your entries:

Module	**IDOC_INPUT_ORDERS**
Basic type	`ORDERS01`
Extension	`Z1ORDERS`
Message type	`ORDERS`
Object type	`BUS2032` (sales order)

Checking Partner Profiles (Inbound)

Prerequisites

You must have completed the required steps in Extending an IDoc Type [Page 172] .

Procedure

Your extended function module is addressed in partner profiles through a Process code [Page 71] . All partners, whose partner profiles contain this process code (and the corresponding message type) process the data of this extension in the system. If you want to avoid this with particular partners, then you must create a copy of this (originally not extended) function module and use a process code which refers to the copy.

For partner profile, please refer to the section:

Partner Profiles in the Standard Dialog [Page 74]

In the example, the following fields in the inbound partner profiles are maintained:

Field	Value
Partner number	**TESTCUST**
Partner type	**KU**
Message type	**ORDERS**
Processing	*immediately*
Permitted agent	Your SAP user, type **US** (individual user), language **EN**
Process code	**ORDE**

Testing Inbound Processing

Prerequisites

You must have completed the required steps for inbound processing in:

- Extending an IDoc Type [Page 172] (if you have extended a basic type) **or**
- Defining and Using a New Basic Type [Page 189] (if you have defined a new basic type)

Procedure

1. You should use the inbound test tool [Page 96] , as IDocs of the relevant type can be edited directly. Select the following as test templates:
   - An outbound IDoc, if you have configured and tested outbound processing for your IDoc type. You must then exchange the sender and recipient entries and enter 2 as the direction in the control record.
   - Your new IDoc type, if you do not yet have any IDocs of the new type (extended or new basic type). In this case, you must also maintain the segment fields with application data.

     In direct ALE inbound processing, you can access the function module directly and in debugging mode from the test tool. This can be very helpful if new developments are involved. In inbound processing via workflow test your processing, for example, in the ABAP Workbench.
2. Check the current status of the new IDoc using the IDoc display [Page 117] function. If the application has accepted the document, the **current** IDoc status is set to 53 (“application document posted”). Otherwise, the current status is 51 (“application document not posted”). You can display the errors which have been written to the status records by the application by clicking on the records.
3. Check whether the application has written the IDoc data to the correct tables.

   If the test tool displays an error message (for example, “IDoc could not or should not be processed”), you can jump from this message to the IDoc display directly.

   In the example, you should make the following entries in the control record:

Recipient:	Port SAPC11, partner number C11CLNT<current client>, partner type LS (logical system), leave the partner function blank
Sender:	Port TESTPORT, partner number TESTCUST, partner type KU (customer), partner function AG (SP)

If you have extended the basic type ORDERS01:

IDoc type	ORDERS01
Extension	Z1ORDERS

Message type	ORDERS

If you have defined the new basic type TESTER01:

IDoc type	TESTER01
Message type	TESTER

Releasing New Objects

Prerequisites

You must have completed the required steps in:

- Extending an IDoc Type [Page 172] (if you have extended a basic type) **or**
- Defining and Using a New Basic Type [Page 189] (if you have defined a new basic type)

Procedure

You should release the new segments first, then your basic type or extension.

You can set the release from the relevant development tool (IDoc type editor or segment editor) by selecting *Edit → Set release.* The release can also be reset by selecting *Edit → Cancel release.*

In the example, you should release the following development objects in the specified order when testing has been completed successfully:

If you have extended the basic type ORDERS01:

Segment	Z1TEST1
Extension	Z1ORDERS

If you have defined the new basic type TESTER01:

Segments	E1HEAD, E1ITEM
IDoc type	TESTER01

Result

Your new definition is now closed. If you still want to translate the documentation into your new definition, you can do this specifically with the general translation transaction. The following section deals with the objects that you need within the translation transaction:

Documentation translation [Page 151]

Defining and Using a New Basic Type

Purpose

An SAP development creates a new basic type, which represents a new business process within which data can be exchanged electronically.

Prerequisites

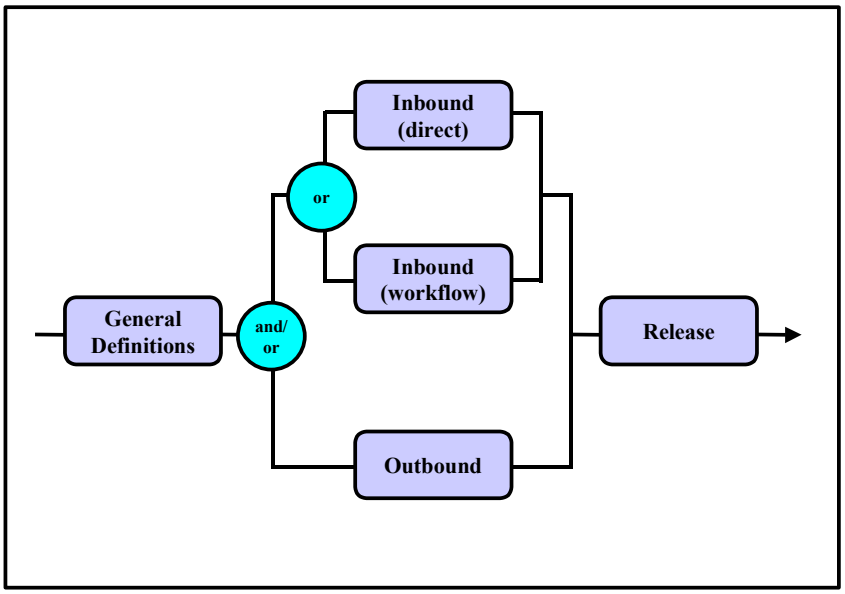
You require corresponding authorizations (authorization object S_IDOCDEFT, for example, in the role SAP_BC_SRV_EDI_DEVELOPER_AG) for the developments. Development takes place in an SAP system.

Process flow

The following subsections contain the individual steps. The list can be used as a checklist when defining the new basic type. More information about the required steps and any alternatives can be found in the process flow diagram.

With the help of an **example**, you can define an actual IDoc type and process the type in a number of different ways. The example is used in each individual step. As a prerequisite, master data for the test customer TESTCUST (partner type KU) and the test vendor TESTVEND (partner type LI) must be present in the system. In addition, the example also assumes that the file port TESTPORT has been created.

Process flow diagram



Individual steps

General Definitions [Page 192]

Combining segments and defining new segments [Page 193]

Defining basic types [Page 195]

Optional: Creating Messages. Mandatory: Assigning Messages to IDoc Types [Page 196]

Creating Object Types [Page 197]

Configuring Outbound Processing (with Message Control) [Page 198]

Writing Function Modules (Outbound) [Page 199]

Creating and Assigning Process Codes [Page 205]

Defining a Partner [Page 206]

Testing Outbound Processing for the New Basic Type [Page 207]

Configuring inbound processing

Configuring Direct Inbound Processing (with ALE) [Page 208]

Creating a New Function Module [Page 209]

Maintaining the Attributes of a Function Module [Page 220]

Assigning a Function Module (Direct Inbound Processing) [Page 221]

Assigning a Process Code (Direct Inbound Processing) [Page 222]

Defining a Partner (Direct Inbound Processing) [Page 224]

General Definitions

Defining Segments

This section involves a scenario in which you have to create a new segment. If you have to create a new segment definition, change an existing segment [Page 164] using the segment editor.

Prerequisites

You must have authorization to create development objects and transport requests.

Procedure

7. Call the segment editor from the Initial Node of the IDoc Interface [Ext.] by selecting *Development → IDoc segments.* Select *Utilities → Requests (organizer)* to create a request.
8. Specify a segment type [Page 156] and select .
9. Specify a description of the segment and enter the field names and data elements: Decide whether your field is to store data which is coded according to the ISO standard (checkbox).

   The field lengths (in bytes) and any fixed values are derived from the data elements via the domains (with field lengths as *export lengths*). The documentation for the data element is also copied and displayed by the documentation tools.

   Do not change the export length. Data is sent to and from the external systems as *character* fields with this length. However, internally (in the IDoc), the data is stored as a field of the same type as the selected data element (for example as an *integer*) with the corresponding length. Different lengths can lead to a loss of data (superfluous bytes are cut off).
10. Choose *Goto → Documentation* and describe the **segment type**. Save the active version of your **application documentation** and return to the segment editor.

    If you use the segment later in an IDoc type and display this type via the documentation tools, the description you have entered here will be displayed.
11. Save the segment definition [Page 156] by selecting . Enter the segment in a transport request.

    You should assign all your development objects (segments, IDoc type, function group, and so on) for IDoc types to the same development class, so that the transport properties are always identical.
12. Exit the screen (F3).

    The segment definition you have created appears with the version number 000 in the segment editor. It is also included in the transport request you created.

Result

You have now created a new segment type with one segment definition. Although not released [Page 158], you can still test and work with the segment type. **You should only release the segment after you have successfully tested the processing.**

In the example, you should define two segments - E1HEAD (header) and E1ITEM (item). These two segments already exist in the system. They are only used as examples and should not be used in live systems. For example, the segment E1HEAD starts with the following fields:

Field	Data element	Length
BELNR (document no.)	EDI_BELNR	35
VKORG (sales org.)	VKORG	4
VTWEG (distrib. channel)	VTWEG	2
SPART (division)	SPART	2
...	...	...

Defining a Basic Type

Prerequisites

You must have authorization to create development objects and transport requests.

Procedure

1. Go into the initial screen of the IDoc type editor: In the Initial Node of the IDoc Interface [Ext.] choose *Development* → *IDoc types*.
2. Enter a name for the basic type and select *Basic type* as the development object. Choose . From the following screen, choose for example, *Create new*. Enter a description and choose .
3. Position the cursor on the header and choose .

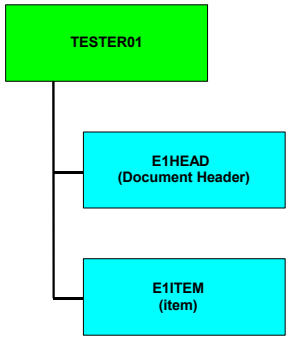
   The attribute maintenance dialog box is displayed.
4. Select a segment. Enter both the maximum and minimum number and select *Mandatory segment*. The segment now appears as the first segment in the new basic type.
5. Create the other segments accordingly, at the same level or as child segments.
6. Save your basic type.

   You should assign all your development objects (segments, IDoc type, function group, and so on) for IDoc types to the same development class, so that the transport properties are always identical.

Result

You have now defined a new basic type. Although not released [Page 158], you can still test and work with the segment type. **You should only release the basic type after you have tested the processing successfully.**

In the example, your IDoc type is called TESTER01 and has the following structure (all mandatory segments, maximum number = minimum number = 1):



Assigning Basic Types to Message Types

The business background is established by assigning the IDoc type to one (or more) messages(s). For example, the “logical message type” *ORDERS* is usually used for the purchase order in outbound processing and for the order in inbound processing.

Since an IDoc type can be assigned to several logical message types and vice versa, you should only **create** a new message if none of the existing messages adequately describes your new business process.

Procedure

Creating a new message type (“logical message”) (optional)

1. From the Initial Node of the IDoc Interface [Ext.] choose *Development* → *Message Types*.
2. Select and then *New Entries*.
3. Enter the new message type with a short description and save your entries. When saving the message, place this development object in your change request for your basic type definition.

Assigning basic types to message types (required)

4. From the initial node of the IDoc interface, select *Development* → *IDoc type/message*.
5. Select and then *New Entries*.
6. Enter your message type and the basic type in the corresponding columns. Enter the SAP release. Use your change request for the basic type definition again when you save the data.

You can also define these assignments from the *Environment* menu in the IDoc type editor.

Result

You have now assigned a logical message, and therefore a **business significance**, to the IDoc type. The **functional significance** is implemented in the processing via a function module or via a workflow task.

Make the following entries for the example:

Message type (logical message)	TESTER
Basic type	TESTER01

Defining Object Types

Before you can use your new basic type in workflow (optional in inbound processing but required in error handling), you should create a corresponding object type in the *Business Object Repository* (BOR). The object type **inherits** all of the attributes, methods and events of the super type IDOCAPPL. You could also take this object type and only define different methods for exception handling, for example. According to the SAP conventions, however, each message type also corresponds to a BOR object type.

Procedure

1. Call the Workflow area menu from the initial R/3 screen by selecting *Tools* → *SAP Business Workflow* → *Development* Then select *Business Object Builder*.
2. Create the new object type as a subtype of IDOCAPPL. You should therefore enter IDOCAPPL in the field and choose *Subtype*.
3. Enter the attributes for the new object type (name, short description and so on) and a new program name (the program will be generated): The program describes the object type (attributes and methods) in the ABAP language.
4. Select and save the object.
5. Choose .
6. Before you can use the object type, the release status must at least be set to “implemented”. You can assign this status from the initial screen of the Business Object Builder by selecting *Object type* → *Change release status to* → *Implemented.*

See also: BC → Business Management → SAP Business Workflow → Maintaining Object Types [Ext.].

Result

You have defined an object type for your new IDoc.
The two events *inputErrorOccurred* and *inputFinished,* which you have assigned to this object type, are important for exception handling.

In the example, your object type has the following attributes:

Object type	IDOCTEST
Object name	IDOCTEST
Name	Test IDoc
Short description	Test error handling TESTER01
Application	**V** (Sales and Distribution)

Configuring Outbound Processing (with Message Control)

Prerequisite

Your application must be compatible with Message Control.

You have carried out the steps in General definitions [Page 192] .

Procedure

Creating a New Function Module (Outbound Processing under MC) [Page 199]

Assigning a New Process Code to a New Function Module and Message (Outbound Processing) [Page 205]

Defining Partner Profiles for the New IDoc Type (Outbound Processing) [Page 206]

Testing Outbound Processing for the New Basic Type [Page 207]

Creating a Function Module (Outbound Processing Under MC)

This section describes how to create a function module which is identified by the report RSNASTED (form routine NEW_DYN_PERFORM) via a new process code. This function module enters application data in the new basic type. RSNASTED can be found in table TNAPR as an EDI processing program. Message Control reads this table and can then call RSNASTED.

Procedure

1. Use the Function Builder to create a new function module.
2. Create the segments as global data in your function group. The function module must transfer the application data from the application tables to the corresponding segments.
3. Activate the function module: In the initial screen of the Function Builder select .

In the example, the new function module is called IDOC_OUTPUT_TESTER. The function module uses the new interface (from Release 3.0 onwards), that is, the application data is stored in an internal table. In addition, the path without the ALE layer (that is, without filters) is selected.

The function module is called from Purchasing (create purchase order) in the same way as IDOC_OUTPUT_ORDERS (IDoc type ORDERS01). The function module therefore fills the segments of your new IDoc type from a purchasing table, that is, EKKO and EKPO (purchasing document header and item). Note that the intention here is not to simulate a realistic purchase order, but to describe how data reaches an **IDoc table** from **application tables** via **segment structures**. In this case, you should pay particular attention to the form routines HEADER_E1HEAD_FILL or POSITION_E1ITEM_FILL.

Administration parameters for IDOC_OUTPUT_TESTER

Application abbreviation	M (Materials Management)
Processing type	normal, start immediately

Interface for IDOC_OUTPUT_TESTER

Formal parameters	**Reference structure**	**Explanation**
Import parameters		
OBJECT	MC	Current Message Control record
CONTROL_RECORD_IN	EDIDC	Contains recipient information
Export parameters		
OBJECT_TYPE	WFAS1-ASGTP	Object type for the application object in the *Business Object Repository* (BOR)

Creating a Function Module (Outbound Processing Under MC)

CONTROL_RECORD_OUT	EDIDC	Contains the sender information as well as the date and time at which the IDoc table was filled.
Table		
INT_EDIDD	EDIDD	IDoc data records (internal table)

Coding example

```
FUNCTION IDOC_OUTPUT_TESTER.
*"----------------------------------------------------------------
*"*"Global interface:
*"       IMPORTING
*"             VALUE(OBJECT) LIKE  NAST STRUCTURE  NAST
*"             VALUE(CONTROL_RECORD_IN) LIKE  EDIDC STRUCTURE  EDIDC
*"       EXPORTING
*"             VALUE(OBJECT_TYPE) LIKE  WFAS1-ASGTP
*"             VALUE(CONTROL_RECORD_OUT) LIKE  EDIDC STRUCTURE  EDIDC
*"       TABLES
*"             INT_EDIDD STRUCTURE  EDIDD
*"       EXCEPTIONS
*"             ERROR_MESSAGE_RECEIVED
*"----------------------------------------------------------------------

  CLEAR CONTROL_RECORD_OUT.
* fill help fields
  H_KAPPL = OBJECT-KAPPL.
H_EBELN = OBJECT-OBJKY.
  H_PARVW = OBJECT-PARVW.
* fill control record
  MOVE CONTROL_RECORD_IN TO CONTROL_RECORD_OUT.
  CONTROL_RECORD_OUT-DIRECT = '1'.
  CONTROL_RECORD_OUT-SERIAL = SY-DATUM.
  CONTROL_RECORD_OUT-SERIAL+8 = SY-UZEIT.
* read orders
  PERFORM ORDERS_READ.
* fill idoc table
  PERFORM IDOC_TABLE_FILL.
* provide object type
  OBJECT_TYPE = 'BUS2012'.

ENDFUNCTION.


FORM ORDERS_READ.

  SELECT SINGLE * FROM T000 WHERE MANDT EQ SY-MANDT.

* read header data
  SELECT SINGLE * FROM EKKO WHERE EBELN EQ H_EBELN.
  IF SY-SUBRC NE 0.
    MESSAGE E751 WITH H_EBELN.
  ENDIF.
```

```
  EDIDC-SNDPRT = 'MM'.
  EDIDC-SNDPRN = EKKO-EKORG.

* read data of supplier (book keeping view)
  SELECT SINGLE * FROM LFB1 WHERE LIFNR = EKKO-LIFNR
                              AND BUKRS = EKKO-BUKRS.
* fill purchasing organization
  IF LFB1-EIKTO EQ SPACE.
    LFB1-EIKTO = EKKO-EKORG.
  ENDIF.
* read position data
  SELECT * FROM EKPO WHERE EBELN = EKKO-EBELN.
    MOVE-CORRESPONDING EKPO TO XEKPO.
    APPEND XEKPO.
  ENDSELECT.

* read schedule lines
  SELECT * FROM EKET WHERE EBELN = EKKO-EBELN.
  ENDSELECT.

ENDFORM.                                " ORDERS_READ


FORM IDOC_TABLE_FILL.

* header data
  PERFORM HEADER_E1HEAD_FILL.
* data in position
  PERFORM POSITION_E1ITEM_FILL.

ENDFORM.                                 " IDOC_TABLE_FILL


FORM HEADER_E1HEAD_FILL.

  CLEAR INT_EDIDD.
  CLEAR E1HEAD.
  INT_EDIDD-SEGNAM = 'E1HEAD'.

* fill fields
* document number
  E1HEAD-BELNR = EKKO-EBELN.

* The following constants (assignment of customer/vendor to sales
organization) are only written in the segments, so that the subsequent
inbound processing works correctly in the example for SD. In the case
of a "real" standard order in SD inbound processing, Customizing tables
are maintained or the assignment is transferred from the EDI subsystem.

* sales organization (not supplied -> constant)
    E1HEAD-VKORG = '1000'.
* distribution channel (not supplied -> constant)
    E1HEAD-VTWEG = '10'.
```

```
* division (not supplied -> constant)
    E1HEAD-SPART = '00'.
  ENDSELECT.
* order type
E1HEAD-AUART = 'NB'.
* date of delivery
  IF EKET-EINDT NE 0.
    E1HEAD-WLDAT = EKET-EINDT.
  ELSE.
    E1HEAD-WLDAT = SY-DATUM + 21.
  ENDIF.
  CONDENSE E1HEAD-WLDAT.
* ordering party / sold to party (AG)
  E1HEAD-AUGEB = LFB1-EIKTO.
* supplier
  E1HEAD-LIEF  = EKKO-LIFNR.
* order date
  E1HEAD-BSTDK = EKKO-BEDAT.
  CONDENSE E1HEAD-BSTDK.
* order time
  E1HEAD-BELUZT = SY-UZEIT.

* move data to segment area of data record
  MOVE E1HEAD TO INT_EDIDD-SDATA.
* append data record to internal table
  APPEND INT_EDIDD.

* customer function in order to change header segment
  CALL CUSTOMER-FUNCTION '001'
       EXPORTING
            PI_EKKO              = EKKO
       IMPORTING
            PE_EKKO              = EKKO
       TABLES
            PT_IDOC_DATA_RECORDS = INT_EDIDD.

ENDFORM.                               " HEADER_E1HEAD_FILL


FORM POSITION_E1ITEM_FILL.

* loop at positions
  LOOP AT XEKPO.

    CLEAR INT_EDIDD.
    CLEAR E1ITEM.
    INT_EDIDD-SEGNAM = 'E1ITEM'.
    MOVE XEKPO TO EKPO.

* fill fields
* position number
    E1ITEM-POSEX = EKPO-EBELP.
    CONDENSE E1ITEM-POSEX.
```

```
* material number
    IF EKPO-MATNR NE SPACE.
       E1ITEM-MATNR = EKPO-MATNR.
    ENDIF.

* amount
    E1ITEM-MENGE = EKPO-MENGE.
    CONDENSE E1ITEM-MENGE.
* unit
    PERFORM ISO_CODE_UNIT USING EKPO-MEINS.
    E1ITEM-BMEINH = EKPO-MEINS.
* material number of supplier
    IF EKPO-IDNLF NE SPACE.
       E1ITEM-LMATNR =  EKPO-IDNLF.
    ENDIF.

* move data to segment area of data record
    MOVE E1ITEM TO INT_EDIDD-SDATA.
* append data record to internal table
    APPEND INT_EDIDD.

* customer function for position segment
  CALL CUSTOMER-FUNCTION '002'
       EXPORTING
            PI_EKPO              = EKPO
       IMPORTING
            PE_EKPO              = EKPO
       TABLES
            PT_IDOC_DATA_RECORDS = INT_EDIDD.

ENDFORM.                               " HEADER_E1HEAD_FILL

  ENDLOOP.

ENDFORM.                                " POSITION_E1ITEM_FILL


FORM ISO_CODE_UNIT USING  ICU_UNIT.

  CHECK ICU_UNIT NE SPACE.
  CALL FUNCTION 'UNIT_OF_MEASURE_SAP_TO_ISO'
       EXPORTING
            SAP_CODE    = ICU_UNIT
       IMPORTING
            ISO_CODE    = ISO_UNIT
       EXCEPTIONS
            NOT_FOUND   = 01
            NO_ISO_CODE = 02.

  IF SY-SUBRC NE 0.
    MESSAGE I764 WITH EKPO-EBELP ICU_UNIT.
  ENDIF.
```

Creating a Function Module (Outbound Processing Under MC)

```
  MOVE ISO_UNIT TO ICU_UNIT.

ENDFORM.                                " ISO_CODE_UNIT
```

The HLEVEL field (hierarchy level of segment) in the administration section is not filled. This is carried out by the IDoc interface, which receives this value from the definition of the IDoc type TESTER01.

You should also note that the segment in the INT_EDIDD-SEGNAM field must be written in **upper case letters**. Otherwise, the IDoc interface will return a syntax error.

Global data from IDOC_OUTPUT_TESTER

```
*- Tabellen ------------------------------------------------------*
TABLES:
        EKPO,
        EKET,
        LFB1,
        EDIDC,
        E1HEAD,
        E1ITEM,
        T000,
        EDSDC.
* help fields
DATA:
        H_EBELN LIKE EKKO-EBELN,
        H_KAPPL LIKE NAST-KAPPL,
        H_PARVW LIKE EKPA-PARVW.
* iso codes
DATA:
        ISO_UNIT  LIKE T006-ISOCODE.

TYPE-POOLS ISOC.
*- Direct values for Return_variables ---------------------------------
DATA:
EID LIKE BDWFRETVAR-WF_PARAM VALUE 'Error_IDOCs',
        PID LIKE BDWFRETVAR-WF_PARAM VALUE 'Processed_IDOCs',
        APO LIKE BDWFRETVAR-WF_PARAM VALUE 'Appl_Objects',
        APE LIKE BDWFRETVAR-WF_PARAM VALUE 'Appl_Object'.

*- Help fields for change document ------------------------------------*
  INCLUDE FM06ECDT.

*- Common part for change document ------------------------------------*
  INCLUDE FM06LCCD.

*- Direct values ------------------------------------------------------*
  INCLUDE FMMEXDIR.
```

Assigning a Process Code (Outbound Processing)

The outbound function module is triggered via a process code in the IDoc interface.

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

1. From the Initial Node of the IDoc Interface [Ext.] choose *Control* → *Outbound Process Code*. Choose the *outbound process code* option.
2. Select and *New Entries*.
3. Enter a process code, a description and the function module. Choose a version of the function module and select whether the ALE services are to be called.
4. Save your entries. Choose the navigation *Logical Message.*
5. Choose *New entries.*
6. Enter the logical message and the process code in the corresponding fields and save your entries. As a result, the value help (F4) in the partner profiles can display the possible process codes from the logical message, that is, the business view.

Result

You have assigned a new process code for the function module. You can now enter this process code in the partner profiles.

Make the following entries for the example:

Process code	TESTOUT
Function module	IDOC_OUTPUT_TESTER
Message type (logical message)	TESTER
Option ALE/inbound	*Processing without ALE*
Function module version	*Processing with function module version 3.0*

Defining a Partner (Outbound Processing)

The new basic type must be entered in the partner profiles before being used. In the case of outbound processing under Message Control, enter the new process code, which identifies the new outbound function module, for the basic type. For more information about the procedure, see: Partner Profiles in the Standard Dialog [Page 74]

In the example, you should maintain the following fields in the partner profiles for “General outbound” and “Outbound processing using Message Control”:

Field	Value
Partner number	**TESTVEND**
Partner type	**LI**
Partner function	**VD**
Message type	**TESTER**
IDoc type	**TESTER01**
Receiver port	**TESTPORT**
Output mode	*Transfer IDocs immediately*, *Do not start subsystem*
Permitted agent	Your SAP user, type **US** (individual user), language **EN**
Application	**EF** (purchasing)
Output type	**NEU (new)**
Process code	**TESTOUT**

Testing Outbound Processing

Prerequisites

You must have completed the required steps in:

- Extending an IDoc Type [Page 172] (if you have extended a basic type) **or**
- Defining and Using a New Basic Type [Page 189] (if you have defined a new basic type)

Procedure

The IDoc is generated from the application. If you have configured the Dispatch Time [Ext.] as “1” in the condition record for outbound processing under Message Control (MC), only the MC record is generated. You can test the MC record using the program Test from MC [Page 97]

In the example, create a purchase order (transaction ME21). The recipient is TESTVEND, you are ordering a material from the material master. In the condition record (processing in the SAP purchasing transaction: *Header* → *Messages*) you have selected dispatch time 4 (processing via *additional specifications)*, that is, the IDoc is generated immediately. In segment Z1TEST1 it shows the value “Europe“ in field CONTINENT: You can check this using IDoc Display [Page 117].

Configuring Direct Inbound Processing (with ALE)

You can configure direct inbound processing which **always** runs via the ALE services.

Prerequisites

General Definitions [Page 192]

Procedure

Creating a Function Module (Direct Inbound Processing) [Page 209]
Maintaining the Attributes of a Function Module [Page 220]
Assigning a Function Module (Direct Inbound Processing) [Page 221]
Assigning a Process Code (Direct Inbound Processing) [Page 222]
Defining a Partner (Direct Inbound Processing) [Page 224]
Creating a Standard Task (Exception) [Page 239]
Testing Inbound Processing [Page 242]
Testing Exception Handling [Page 244]

Creating a Function Module (Direct Inbound Processing)

This step describes how to create a function module which is identified by the IDoc interface via a new process code and called from ALE (field TBD52-FUNCNAME). Direct inbound processing via a function module (not via a workflow) **always** includes the ALE layer. This setting (processing with function module and ALE layer) is identified by the value 6 in the field TEDE2-EDIVRS, which is read by the function module IDOC_START_INBOUND. IDOC_START_INBOUND then calls ALE.

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

1. Use the Function Builder initial screen to create a new function module.
2. Create the segments as global data in your function group. The function module should copy the application data from the segments into the corresponding application tables and modify the IDoc status accordingly. If an error occurs, the function module must set the corresponding workflow parameters for exception handling.
3. Activate the function module: From the initial screen of the Function Builder select  .

In the example, create the function module IDOC_INPUT_TESTER with a global interface. The function module is called when an IDoc of type TESTER01 is received for inbound processing. You will assign an application object (“standard order”) to this IDoc type and therefore maintain tables from SD. To do this, call transaction VA01 using the command CALL TRANSACTION. Please note that the intention here is not to simulate a realistic standard order, but only to illustrate how data reaches **application tables** from an **IDoc table** via **segment structures** (form routine READ_IDOC_TESTER) and how the function module triggers an event for exception handling (by returning suitable return variables to the ALE layer in the form routine RETURN_VARIABLES_FILL).

You can find a comprehensive example of coding for an inbound function module in the ALE documentation in the R/3 Library under Cross application functions → Business Framework Architecture → Application Link Enabling → ALE Programming → Inbound processing → Function module in inbound processing → Processing individual IDocs → Coding example [Ext.]. This function module, for example, also checks whether the logical message is correct and calls a (fictitious) second function module which first writes the application data and then returns the number of the generated document. In addition, status 53 is only set if the application document was posted correctly.

Administration parameters for IDOC_INPUT_TESTER

Application abbreviation	**V** (Sales and Distribution)
Processing type	normal, start immediately

Creating a Function Module (Direct Inbound Processing)

Interface for IDOC_INPUT_TESTER (global interface)

Formal parameters	Reference structure	Explanation
Import parameters		
INPUT_METHOD	BDWFAP_PAR-INPUTMETHD	Describes how the function module is to be processed (example: in the background)
MASS_PROCESSING	BDWFAP_PAR-MASS_PROC	Mass inbound processing? (flag)
Export parameters		
WORKFLOW_RESULT	BDWFAP_PAR-RESULT	Set to 99999 if a workflow event is to be triggered for error handling
APPLICATION_VARIABLE	BDWFAP_PAR-APPL_VAR	Variable freely available from application for workflow
IN_UPDATE_TASK	BDWFAP_PAR-UPDATETASK	Asynchronous update? (flag is not set in example)
CALL_TRANSACTION_DONE	BDWFAP_PAR-CALLTRANS	Transaction called? (flag is not set in example)
Table		
IDOC_CONTRL	EDIDC	IDoc control record
IDOC_DATA	EDIDD	IDoc data records
IDOC_STATUS	BDIDOCSTAT	IDoc status records for ALE
RETURN_VARIABLES	BDWFRETVAR	IDoc assigned to workflow method parameters
SERIALIZATION_INFO	BDI_SER	If several IDocs are to be processed in a certain sequence: this structure contains the necessary information

Coding example

```
FUNCTION IDOC_INPUT_TESTER.
*"----------------------------------------------------------------------
*"*"Global interface:
*"        IMPORTING
*"              VALUE(INPUT_METHOD) LIKE  BDWFAP_PAR-INPUTMETHD
*"              VALUE(MASS_PROCESSING) LIKE  BDWFAP_PAR-MASS_PROC
```

```
*"       EXPORTING
*"             VALUE(WORKFLOW_RESULT) LIKE  BDWFAP_PAR-RESULT
*"             VALUE(APPLICATION_VARIABLE) LIKE  BDWFAP_PAR-APPL_VAR
*"             VALUE(IN_UPDATE_TASK) LIKE  BDWFAP_PAR-UPDATETASK
*"             VALUE(CALL_TRANSACTION_DONE) LIKE  BDWFAP_PAR-CALLTRANS
*"       TABLES
*"              IDOC_CONTRL STRUCTURE  EDIDC OPTIONAL
*"              IDOC_DATA STRUCTURE  EDIDD
*"              IDOC_STATUS STRUCTURE  BDIDOCSTAT
*"              RETURN_VARIABLES STRUCTURE  BDWFRETVAR
*"              SERIALIZATION_INFO STRUCTURE  BDI_SER
*"----------------------------------------------------------------------

* initialize SET/GET Parameter and internal tables
  PERFORM INITIALIZE_ORGANIZATIONAL_DATA.
* Move IDOC to internal tables of application
 PERFORM READ_IDOC_TESTER.
* call transaction Order Entry VA01
 PERFORM CALL_VA01_IDOC_ORDERS USING ERRORCODE.
* set status value
  perform write_status_record using errorcode.
* return values of function module
  PERFORM RETURN_VARIABLES_FILL USING ERRORCODE.

ENDFUNCTION.


FORM INITIALIZE_ORGANIZATIONAL_DATA.

* initialize SET/GET parameters
   SET PARAMETER ID 'VKO' FIELD SPACE.
   SET PARAMETER ID 'VTW' FIELD SPACE.
   SET PARAMETER ID 'SPA' FIELD SPACE.
   SET PARAMETER ID 'VKB' FIELD SPACE.
   SET PARAMETER ID 'VKG' FIELD SPACE.

* initialize internal tables
   REFRESH BDCDATA.
   CLEAR BDCDATA.
   CLEAR BELEGNUMMER.
   CLEAR ERRTAB.
   REFRESH ERRTAB.
   REFRESH XBDCMSGCOLL.
   CLEAR XBDCMSGCOLL.

ENDFORM.                    " INITIALIZE_ORGANIZATIONAL_DATA

FORM READ_IDOC_TESTER.

  PERFORM INITIALIZE_IDOC.
 LOOP AT IDOC_DATA
   WHERE DOCNUM = IDOC_CONTRL-DOCNUM.
    CASE IDOC_DATA-SEGNAM.
```

```
* header data
      WHEN 'E1HEAD'.
        MOVE IDOC_DATA-SDATA TO E1HEAD.
        PERFORM PROCESS_SEGMENT_E1HEAD.
* position data
      WHEN 'E1ITEM'.
        MOVE IDOC_DATA-SDATA TO E1ITEM.
        PERFORM PROCESS_SEGMENT_E1ITEM.
    ENDCASE.
 ENDLOOP.
* only when there were one or more items
  CHECK FIRST NE 'X'.
  APPEND XVBAP.                        "last one

ENDFORM.                    " READ_IDOC_TESTER

FORM INITIALIZE_IDOC.

  CLEAR XVBAK.
  REFRESH XVBAP.
  CLEAR XVBAP.
  POSNR = 0.
  FIRST = 'X'.

ENDFORM.                    " INITIALIZE_IDOC

FORM PROCESS_SEGMENT_E1HEAD.

* requested date of delivery
  WLDAT = E1HEAD-WLDAT.
* delivery date
  XVBAK-BSTDK = E1HEAD-BSTDK.
* customer number
  XVBAK-KUNNR = E1HEAD-AUGEB.
* order number
  XVBAK-BSTNK = E1HEAD-BELNR.
* division
  XVBAK-SPART = E1HEAD-SPART.
* distribution channel
  XVBAK-VTWEG = E1HEAD-VTWEG.
* sales organization
  XVBAK-VKORG = E1HEAD-VKORG.
* order type
  XVBAK-AUART = E1HEAD-AUART.
* do not fill incoterms (inco1, inco2)
* customer function
  CALL CUSTOMER-FUNCTION '001'
       EXPORTING
            PI_VBAK621           = XVBAK
       IMPORTING
            PE_VBAK621           = XVBAK
       TABLES
            PT_IDOC_DATA_RECORDS = IDOC_DATA.
```

```
ENDFORM.                    " PROCESS_SEGMENT_E1HEAD

FORM PROCESS_SEGMENT_E1ITEM.
* position number
  XVBAP-POSNR = XVBAP-POSNR + 1.
* amount
  XVBAP-WMENG = E1ITEM-MENGE.
* unit
  CALL FUNCTION 'ISO_TO_SAP_MEASURE_UNIT_CODE'
       EXPORTING
            ISO_CODE  = E1ITEM-BMEINH
       IMPORTING
            SAP_CODE  = XVBAP-VRKME
       EXCEPTIONS
            OTHERS    = 0.
* material number
  XVBAP-MATNR = E1ITEM-LMATNR.

CALL CUSTOMER-FUNCTION '002'
       EXPORTING
            PI_VBAP621           = XVBAP
       IMPORTING
            PE_VBAP621           = XVBAP
       TABLES
            PT_IDOC_DATA_RECORDS = IDOC_DATA.
APPEND XVBAP.

ENDFORM.                    " PROCESS_SEGMENT_E1ITEM

FORM CALL_VA01_IDOC_ORDERS USING ERRORCODE.

* call transaction first dynpro
  PERFORM DYNPRO_START.
* call transaction double-line entry
  PERFORM DYNPRO_DETAIL2.
* incoterms
  PERFORM DYNPRO_HEAD_300.
* call transaction item datas
  PERFORM DYNPRO_POSITION.
  PERFORM DYNPRO_SET USING 'BDC_OKCODE' 'SICH'.
* determine input method
  IF INPUT_METHOD IS INITIAL.
    INPUT_METHOD = 'N'.
  ENDIF.
* call transaction VA01
 CALL TRANSACTION 'VA01' USING    BDCDATA
                         MODE     INPUT_METHOD
                         UPDATE   'S'
                         MESSAGES INTO XBDCMSGCOLL.


errorcode = SY-SUBRC.       " remember returncode for status update
```

```
ENDFORM.                    " CALL_VA01_IDOC_ORDERS


form write_status_record using errorcode.

* FILL IDOC_STATUS
 IDOC_STATUS-DOCNUM = IDOC_CONTRL-DOCNUM.
 IF ERRORCODE = 0.
IDOC_STATUS-STATUS = BELEG_GEBUCHT.      "value 53
   GET PARAMETER ID 'AUN' FIELD BELEGNUMMER.
   IDOC_STATUS-MSGID = 'V1'.
IDOC_STATUS-MSGNO = '311'.
   IDOC_STATUS-MSGV1 = 'Terminauftrag'.
   IDOC_STATUS-MSGV2 = BELEGNUMMER.
ELSE.
    IDOC_STATUS-STATUS = BELEG_NICHT_GEBUCHT.  "value 51
    IDOC_STATUS-MSGID = SY-MSwGID.
    IDOC_STATUS-MSGNO = SY-MSGNO.
    IDOC_STATUS-MSGV1 = SY-MSGV1.
    IDOC_STATUS-MSGV2 = SY-MSGV2.
    IDOC_STATUS-MSGV3 = SY-MSGV3.
    IDOC_STATUS-MSGV4 = SY-MSGV4.
  ENDIF.
  APPEND IDOC_STATUS.

ENDFORM.


FORM DYNPRO_START.

  PERFORM DYNPRO_NEW USING PROGRAMM_AUFTRAG
                           DYNPRO-EINSTIEG
                  CHANGING LAST_DYNPRO.
* ordertype
  PERFORM DYNPRO_SET USING 'VBAK-AUART' XVBAK-AUART.
* sales organization
  PERFORM DYNPRO_SET USING 'VBAK-VKORG' XVBAK-VKORG.
* Distribution channel
  PERFORM DYNPRO_SET USING 'VBAK-VTWEG' XVBAK-VTWEG.
* Division
  PERFORM DYNPRO_SET USING 'VBAK-SPART' XVBAK-SPART.
* Sales office
  PERFORM DYNPRO_SET USING 'VBAK-VKBUR' XVBAK-VKBUR.
* Sales group
  PERFORM DYNPRO_SET USING 'VBAK-VKGRP' XVBAK-VKGRP.

ENDFORM.                     " DYNPRO_START


FORM DYNPRO_NEW USING    PROGNAME
                         DYNPRONR
                CHANGING LAST_DYNPRO.
```

```
 CLEAR BDCDATA.
 BDCDATA-PROGRAM = PROGNAME.
 BDCDATA-DYNPRO  = DYNPRONR.
 BDCDATA-DYNBEGIN   = 'X'.
 APPEND BDCDATA.
 LAST_DYNPRO = DYNPRONR.

ENDFORM.                    " DYNPRO_NEW


FORM DYNPRO_SET USING    FELDNAME
                         FELDINHALT.

  CLEAR BDCDATA.
  CHECK FELDINHALT NE SPACE.
* dynpro field name
  BDCDATA-FNAM = FELDNAME.
* contents
  BDCDATA-FVAL = FELDINHALT.
  APPEND  BDCDATA.

ENDFORM.                    " DYNPRO_SET

FORM DYNPRO_DETAIL2.
* okcode
*  PERFORM DYNPRO_SET USING 'BDC_OKCODE' PANEL-UER2.
* fix dynpro number 4001
  PERFORM DYNPRO_NEW  USING    PROGRAMM_AUFTRAG
                               '4001'
                      CHANGING LAST_DYNPRO.
* order party
  PERFORM DYNPRO_SET      USING 'KUAGV-KUNNR'  XVBAK-KUNNR.
* purchase order number
  PERFORM DYNPRO_SET      USING 'VBKD-BSTKD'   XVBAK-BSTNK.
* requested delivery date
  PERFORM DYNPRO_DATE_SET USING 'VBKD-BSTDK'   XVBAK-BSTDK.
* purchase order date
  PERFORM DYNPRO_DATE_SET USING 'RV45A-KETDAT' WLDAT.

ENDFORM.                    " DYNPRO_DETAIL2

FORM DYNPRO_DATE_SET USING    FIELD NAME
                              FIELD CONTENT.

  DATA: DATE TYPE D.

  CLEAR BDCDATA.
  CHECK FELDINHALT NE SPACE.
  BDCDATA-FNAM = FELDNAME.
  WRITE FELDINHALT  TO DATE.
  BDCDATA-FVAL = DATE.
  APPEND  BDCDATA.
```

```
ENDFORM.                    " DYNPRO_DATE_SET

FORM DYNPRO_HEAD_300.

  PERFORM DYNPRO_SET USING 'BDC_OKCODE' PANEL-KKAU.

* incoterms part 1
  IF NOT XVBAK-INCO1 IS INITIAL.
   PERFORM DYNPRO_SET USING 'VBKD-INCO1' XVBAK-INCO1.
  ENDIF.
* incoterms part 2
  IF NOT XVBAK-INCO2 IS INITIAL.
   PERFORM DYNPRO_SET USING 'VBKD-INCO2' XVBAK-INCO2.
  ENDIF.
*  PERFORM DYNPRO_SET USING 'BDC_OKCODE' 'BACK'.

ENDFORM.                    " DYNPRO_HEAD_300

FORM DYNPRO_POSITION.

  LOOP AT XVBAP.
* dynpro item double line entry
*   PERFORM DYNPRO_SET USING 'BDC_OKCODE' 'UER2'.

    IF XVBAP-POSNR = 1.
* material number
      PERFORM DYNPRO_SET      USING 'VBAP-MATNR(01)'   XVBAP-MATNR.
* order quantity
      PERFORM DYNPRO_SET      USING 'RV45A-KWMENG(01)' XVBAP-WMENG.
* desired delivery date
      PERFORM DYNPRO_DATE_SET USING 'RV45A-ETDAT(1)'  WLDAT.
* sales unit
      PERFORM DYNPRO_SET      USING 'VBAP-VRKME(1)'   XVBAP-VRKME.
    ELSE.
*      PERFORM DYNPRO_SET      USING 'BDC_OKCODE'      'POAN'.

* material number
      PERFORM DYNPRO_SET      USING 'VBAP-MATNR(02)'    XVBAP-MATNR.
* order quantity
      PERFORM DYNPRO_SET      USING 'RV45A-KWMENG(02)'  XVBAP-WMENG.
* desired delivery date
      PERFORM DYNPRO_DATE_SET USING 'RV45A-ETDAT(02)'   WLDAT.
* sales unit
      PERFORM DYNPRO_SET      USING 'VBAP-VRKME(02)'    XVBAP-VRKME.
    ENDIF.
  ENDLOOP.

ENDFORM.                    " DYNPRO_POSITION

FORM RETURN_VARIABLES_FILL USING ERRORCODE.

* allocate IDOC numbers to Workflow output parameters
```

```
  IF MASS_PROCESSING <> SPACE.
    IF ERRORCODE = 0.
      RETURN_VARIABLES-WF_PARAM = PID.
      RETURN_VARIABLES-DOC_NUMBER = IDOC_CONTRL-DOCNUM.
      APPEND RETURN_VARIABLES.
      RETURN_VARIABLES-WF_PARAM = APO.
      RETURN_VARIABLES-DOC_NUMBER = BELEGNUMMER.
      APPEND RETURN_VARIABLES.
      WORKFLOW_RESULT = C_WF_RESULT_OK.
    ELSE.
      RETURN_VARIABLES-WF_PARAM = EID.
      RETURN_VARIABLES-DOC_NUMBER = IDOC_CONTRL-DOCNUM.
      APPEND RETURN_VARIABLES.
      WORKFLOW_RESULT = C_WF_RESULT_ERROR.
    ENDIF.
  ELSE.
    IF ERRORCODE = 0.
      RETURN_VARIABLES-WF_PARAM = APE.
      RETURN_VARIABLES-DOC_NUMBER = BELEGNUMMER.
      APPEND RETURN_VARIABLES.
      WORKFLOW_RESULT = C_WF_RESULT_OK.
    ELSE.
      WORKFLOW_RESULT = C_WF_RESULT_ERROR.
    ENDIF.
  ENDIF.

ENDFORM.                    " RETURN_VARIABLES_FILL

Globale Daten von IDOC_INPUT_TESTER

TABLES: E1HEAD, E1ITEM.

DATA: BEGIN OF BDCDATA OCCURS 500.
        INCLUDE STRUCTURE BDCDATA.
DATA: END OF BDCDATA.

DATA: BEGIN OF XVBAK.                "Kopfdaten
     INCLUDE STRUCTURE VBAK621.
DATA: END OF XVBAK.

DATA: BEGIN OF XVBAP OCCURS 50.        "Position
       INCLUDE STRUCTURE VBAP.
DATA:  WMENG(18) TYPE C.
DATA:  LFDAT LIKE VBAP-ABDAT.
DATA:  KSCHL LIKE KOMV-KSCHL.
DATA:  KBTRG(16) TYPE C.
DATA:  KSCHL_NETWR LIKE KOMV-KSCHL.
DATA:  KBTRG_NETWR(16) TYPE C.
DATA:  INCO1 LIKE VBKD-INCO1.
DATA:  INCO2 LIKE VBKD-INCO2.
DATA:  YANTLF(1) TYPE C.
DATA:  PRSDT LIKE VBKD-PRSDT.
DATA:  HPRSFD LIKE TVAP-PRSFD.
```

```
DATA: END OF XVBAP.

DATA: BEGIN OF DYNPRO,
      EINSTIEG          LIKE T185V-DYNNR VALUE 101,
      KKAU              LIKE T185V-DYNNR,
      UER2              LIKE T185V-DYNNR,
      KBES              LIKE T185V-DYNNR,
      ERF1              LIKE T185V-DYNNR,
      PBES              LIKE T185V-DYNNR,
      PKAU              LIKE T185V-DYNNR,
      PEIN              LIKE T185V-DYNNR,
      EID1              LIKE T185V-DYNNR,
      POPO              LIKE T185V-DYNNR,
      EIPO              LIKE T185V-DYNNR,
      KPAR              LIKE T185V-DYNNR,
      PSDE              LIKE T185V-DYNNR,
      PPAR              LIKE T185V-DYNNR,
      KDE1              LIKE T185V-DYNNR,
      KDE2              LIKE T185V-DYNNR,
      PDE1              LIKE T185V-DYNNR,
      PDE2              LIKE T185V-DYNNR,
      PKON              LIKE T185V-DYNNR,
      END OF DYNPRO.

DATA: BEGIN OF PANEL,
      KKAU              LIKE T185V-PANEL VALUE 'KKAU',
      UER2              LIKE T185V-PANEL VALUE 'UER2',
      KBES              LIKE T185V-PANEL VALUE 'KBES',
      ERF1              LIKE T185V-PANEL VALUE 'ERF1',
      PBES              LIKE T185V-PANEL VALUE 'PBES',
      PKAU              LIKE T185V-PANEL VALUE 'PKAU',
      PEIN              LIKE T185V-PANEL VALUE 'PEIN',
      EID1              LIKE T185V-PANEL VALUE 'EID1',
      EIAN              LIKE T185V-PANEL VALUE 'EIAN',
      POPO              LIKE T185V-PANEL VALUE 'POPO',
      EIPO              LIKE T185V-PANEL VALUE 'EIPO',
      KPAR              LIKE T185V-PANEL VALUE 'KPAR',
      PSDE              LIKE T185V-PANEL VALUE 'PSDE',
      POAN              LIKE T185V-PANEL VALUE 'POAN',
      PPAR              LIKE T185V-PANEL VALUE 'PPAR',
      KDE1              LIKE T185V-PANEL VALUE 'KDE1',
      KDE2              LIKE T185V-PANEL VALUE 'KDE2',
      PDE1              LIKE T185V-PANEL VALUE 'PDE1',
      PDE2              LIKE T185V-PANEL VALUE 'PDE2',
      PKON              LIKE T185V-PANEL VALUE 'PKON',
      KOAN              LIKE T185V-PANEL VALUE 'KOAN',
      END OF PANEL.

DATA: BEGIN OF ERRTAB OCCURS 20,
       TRANS  LIKE TSTC-TCODE,
       ARBGB  LIKE T100-ARBGB,
       CLASS(1) TYPE C,
       MSGNR LIKE T100-MSGNR,
```

```
*      TEXT LIKE T100-TEXT,
       TEXT(123) TYPE C,
       MSGV1 LIKE SY-MSGV1,
       MSGV2 LIKE SY-MSGV2,
       MSGV3 LIKE SY-MSGV3,
       MSGV4 LIKE SY-MSGV4,
      END OF ERRTAB.
*---- Hilfsfelder     ------------------------------------------------*

DATA: PROGRAMM_AUFTRAG LIKE T185V-AGIDV VALUE 'SAPMV45A'.

DATA: LAST_DYNPRO      LIKE T185V-DYNNR,
      WLDAT            LIKE VBAK-BSTDK,
      POSNR            LIKE VBAP-POSNR,
      FIRST(1)         TYPE C VALUE 'X'.

DATA: BEGIN OF XBDCMSGCOLL OCCURS 10.
        INCLUDE STRUCTURE BDCMSGCOLL.
DATA: END OF XBDCMSGCOLL.

* Terminauftrag  ( Auftragsart wird fest gesetzt !)
DATA:   BELEGNUMMER LIKE VBAK-VBELN.
DATA:   ERRORCODE LIKE SY-SUBRC.

* Statuswerte fuer IDOC-Status
DATA:   BELEG_NICHT_GEBUCHT LIKE TEDS1-STATUS VALUE '51'.
DATA:   BELEG_GEBUCHT       LIKE TEDS1-STATUS VALUE '53'.

*- Direktwerte für Return_variables -------------------------
data:
    eid like bdwfretvar-wf_param value 'Error_IDOCs',
    pid like bdwfretvar-wf_param value 'Processed_IDOCs',
    apo like bdwfretvar-wf_param value 'Appl_Objects',
    ape like bdwfretvar-wf_param value 'Appl_Object'.

*- Direktwerte für Workflow_Result --------------------------
DATA: C_WF_RESULT_ERROR LIKE BDWFAP_PAR-RESULT VALUE '99999'.
DATA: C_WF_RESULT_OK    LIKE BDWFAP_PAR-RESULT VALUE '0'.
```

Maintaining the Attributes of a Function Module

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

1. Call the ALE development menu: From the R/3 initial screen, select *Tools → Business Framework → ALE → Development*, *IDoc interface → Inbound processing → Function module → Maintain attributes*, *New entries*.
2. Maintain the attributes for the function module and save your entries. Include the new entries in your transport request.

   For an individual input, configure the attribute *Input type* depending on whether or not the called transaction locks the IDocs and updates the IDoc-Status. For further information on “ALE-capable transactions“ see Using call transaction [Ext.].

   In the example, enter 2 as the *input type* (Individual input with IDoc lock in transaction) and select the option “Dialog possible” (the function module should be able to call a dialog transaction).

Assigning a Function Module (Direct Inbound Processing)

You can specify that the new function module may process the new basic type with a certain message. In addition, you assign an application object from the *Business Object Repository* (BOR) (such as a sales order) to the function module, which can transfer the application data from the IDoc to the application object. This step is the **ALE registration** for the function module.

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

1. From the Initial screen of the IDoc interface [Ext.], choose *Development* → *Message/appl. object,* and then *New entries*.
2. Assign your function module to the various objects. Enter 2 as the direction (inbound). Save the new entries in your transport request.

   In the example, you should assign the following objects to each other:

Module	**IDOC_INPUT_TESTER**, type F (function module)
Basic type	TESTER01
Message type	TESTER
Object type	**BUS2032** (sales order)

Assigning a Process Code (Direct Inbound Processing)

This section describes how to assign a new process code to the new function module. This enables the function module to be identified from the partner profiles defined in the IDoc Interface when an IDoc of the new message type is received.

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

1. From the Initial node of the IDoc interface [Ext.] choose *Control* → *Process code inbound*, Navigation *Process code inbound*, then  , then *New Entries*.
2. Enter your process code and a description. Choose the *Processing with ALE* and *Processing by function module* options and save your entries.

   The maintenance screen is displayed (either automatically or manually), in which you can assign the inbound function module to the process code.
3. Choose *New entries* and enter in the detail screen:
   - Your process code
   - *Module (inbound)* frame: your function module and the maximum number of attempts permitted for posting the application data before exception handling is triggered.
   - *IDoc* frame: Your object type, the trigger event **`inputErrorOccurred`** and the completing event **`inputFinished`**.
   - *Application object* frame: Your object type

   Leave the other fields blank. Save your entries and return to the "process codes inbound" initial screen.
5. Go back via  to maintain the inbound process codes and choose the *Logical message* navigation.
6. In change mode choose *New Entries*.
7. Enter the message type (logical message) and the process code and save your entries. As a result, the value help (F4) in the partner profiles can display the possible process codes from the logical message, that is to say, the business view.

   If you chose *all types* in the last step, the value help displays the process code for all message types. This is also valid for both other options for message type and function.

Result

This assignment has specified a new process code for the function module. You can now enter this process code in the partner profiles.

Make the following entries for the example:

Process code	**TESTINB**
Basic type	**TESTER01**
Function module	**IDOC_INPUT_TESTER**
Maximum number of attempts	**0**
Message type (logical message)	**TESTER**
IDoc object type	**IDOCTEST**
Application object type	**BUS2032** (sales order)

Defining a Partner (Direct Inbound Processing)

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

The new basic type must be entered in the partner profiles before being used. In inbound processing, define the new process code for inbound processing and exception handling for the basic type.

For more information about the procedure, see: Partner Profiles in the Standard Dialog [Page 74]

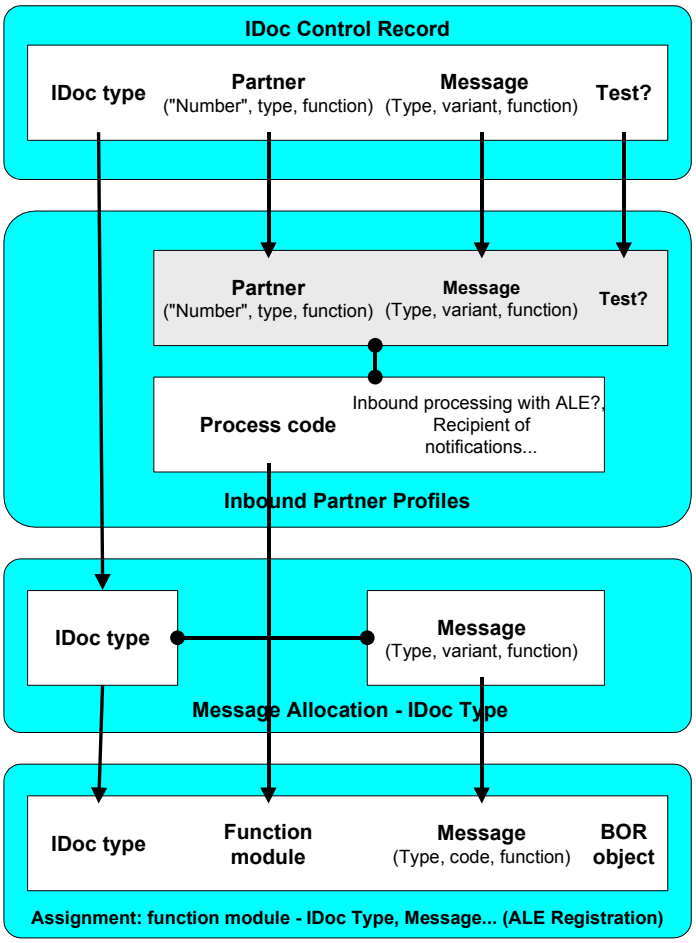
In the example, you should maintain the following fields in the inbound partner profiles (general header entries must exist):

Field	Value
Partner number	**TESTCUST**
Partner type	**KU**
Partner function	**SP**
Message type	**TESTER**
Permitted agent	Your SAP user, type **US** (individual user), language **EN**
Process code	**TESTINB**

Consider the following graphic, which shows what you have defined and the subsequent effect on inbound processing.

The key fields are shown in gray with the assignments made by the R/3 System indicated by arrows. The field definitions shown here result from development of the new basic type and its imbedding in inbound processing. Since you cannot configure the client, this key field is not shown here.

Fields and assignments in inbound processing


IDoc Control Record
IDoc type
Partner
("Number", type, function)
Message
(Type, variant, function)
Test?
Partner
("Number", type, function)
Message
(Type, variant, function)
Test?
Process code
Inbound processing with ALE?,
Recipient of
notifications...
Inbound Partner Profiles
IDoc type
Message
(Type, variant, function)
Message Allocation - IDoc Type
IDoc type
Function
module
Message
(Type, code, function)
BOR
object
Assignment: function module - IDoc Type, Message... (ALE Registration)

Creating a Task (Exception)

In this step, you create a task as the receiver of the event inputErrorOccurred. In addition, you must also activate the type linkage [Ext.].

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

Creating a task by copying

1. From the initial R/3 screen choose *Tools* → *Business Workflow* → *Development, Definition tools* → *Tasks* → *Copy*. Enter *Standard task* as the task type and the ID **00008046** (inbound processing error in ORDERS01).
2. Choose  and enter a task abbreviation. When copying, select the development class which you have configured for yourself in IDoc administration or which applies to all other objects in your basic type definition.

Changing a new task

3. Return to the workflow development menu and choose *Definition tools* → *Task* → *Change*. The system transfers your new task to the input field.
4. Choose  . Replace the object type IDOCORDERS with your new object type for the following tab pages:
   - *Basic data*
   - *Triggering Events* (add new rows, delete old rows)
   - *Terminating Events* (add new rows, delete old rows)

   The object methods *inputForeground* for exception handling are copied from object type IDOCORDERS (tab page *basic data*).
5. Activate the type linkage for *triggering event* (green light) by clicking the button. Save your entries.
6. Select *Additional data* → *Agent assignment* → *Maintain,* to assign possible agents to your task. Via *Attributes* you can classify the task as a general task, that is to say, every user is a possible agent.
7. Save your entries.

You can check whether you have been entered as a possible agent for the task. From the workflow development menu choose *Runtime tools* → *Start workflow (test environment)* and then enter your entries and choose  . If an error message appears which indicates that you are not a possible agent, this may be due to the internal buffer tables used in organization modeling. Then choose *Refresh Organization Environment* and try again.

Make the following entries for the example:

Task abbreviation	**TEST_ERROR**
Object type	**IDOCTEST**
Object method	**errorProcess**
Agent assignment	*general task*

Testing Inbound Processing

Prerequisites

You must have completed the required steps for inbound processing in:

- Extending an IDoc Type [Page 172] (if you have extended a basic type) **or**
- Defining and Using a New Basic Type [Page 189] (if you have defined a new basic type)

Procedure

4. You should use the inbound test tool [Page 96] , as IDocs of the relevant type can be edited directly. Select the following as test templates:
   - An outbound IDoc, if you have configured and tested outbound processing for your IDoc type. You must then exchange the sender and recipient entries and enter 2 as the direction in the control record.
   - Your new IDoc type, if you do not yet have any IDocs of the new type (extended or new basic type). In this case, you must also maintain the segment fields with application data.

   In direct ALE inbound processing, you can access the function module directly and in debugging mode from the test tool. This can be very helpful if new developments are involved. In inbound processing via workflow test your processing, for example, in the ABAP Workbench.
5. Check the current status of the new IDoc using the IDoc display [Page 117] function. If the application has accepted the document, the **current** IDoc status is set to 53 (“application document posted”). Otherwise, the current status is 51 (“application document not posted”). You can display the errors which have been written to the status records by the application by clicking on the records.
6. Check whether the application has written the IDoc data to the correct tables.

   If the test tool displays an error message (for example, “IDoc could not or should not be processed”), you can jump from this message to the IDoc display directly.

   In the example, you should make the following entries in the control record:

Recipient:	Port SAPC11, partner number C11CLNT<current client>, partner type LS (logical system), leave the partner function blank
Sender:	Port TESTPORT, partner number TESTCUST, partner type KU (customer), partner function AG (SP)

If you have extended the basic type ORDERS01:

IDoc type	ORDERS01
Extension	Z1ORDERS

Message type	ORDERS

If you have defined the new basic type TESTER01:

IDoc type	TESTER01
Message type	TESTER

Testing Exception Handling

You can test exception handling by using the test tool to deliberately cause an error, for example sending an IDoc with an empty mandatory field.

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] . You must ensure that your coding calls exception handling (or triggers exception handling in ALE inbound processing via the workflow return parameters) and sets the status value 51.

Procedure

1. From the initial node of the IDoc interface [Ext.] choose *Test* → *Test tool*. Choose *Existing IDoc* as the template for the test and enter the number of an inbound IDoc, which has status 53.
2. Choose  .
3. Use the mouse to click on one of the mandatory segments and delete the contents of the segment.
4. Choose *Standard inbox*. Your IDoc should be assigned error status 51 (you can check using the IDoc display [Page 117] function).
5. If you are entered as a possible agent for your standard task and as a permitted agent in the inbound partner profiles, a work item is sent to your integrated inbox and can be executed from there. Your inbox can be accessed from the IDoc interface initial node by selecting *IDoc* → *Inbox*.

   In the example, you should delete the value from the segment field E1HEAD-VKORG. As a result, an error occurs during inbound processing in the form routine CALL_VA01_IDOC_ORDERS (no sales organization entered). As a possible and selected agent, a work item is sent to your integrated inbox.

Configuring Inbound Processing via Workflow

Prerequisites

General Definitions [Page 192]

Procedure

Defining Inbound Processing as an Object Method [Page 232]

Creating a Standard Task (Inbound Processing via Workflow) [Page 237]

Creating a Process Code (Inbound Processing via Workflow) [Page 238]

Creating a Standard Task (Exception) [Page 239]

Defining a Partner (Inbound Processing via Workflow) [Page 241]

Testing Inbound Processing via Workflow [Page 242]

Testing Exception Handling [Page 244]

Result

You have defined and tested inbound processing (with exception handling) via workflow.

Defining Inbound Processing as an Object Method

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

1. Choose *Tools* → *Business Workflow* → *Development* → *Definition Tools* → *Business Object Builder.* Enter your object type and choose *change*.
2. Define a new method for your object type. Position the cursor on *Methods* and choose .
3. If you have written a function module as the technical implementation of your method, enter the function module as the template. Enter a name for your method and save your entries.

   In the case of *synchronous* methods, you do not have to define an event which is used by the method for confirmations.
4. Position the cursor on the new method and select *Program*.
5. Program the processing.
6. Save the program code and exit the ABAP Editor ( ).
7. Change the method release status in the Business Object Builder to *implemented*: Position the cursor on the method and choose *Edit* → *Change release status* → *Object type component* → *To implemented*.
8. Choose .

Result

You have implemented a method using a specified ABAP program. You must now create a standard task which refers to this method and to the corresponding object type.

In the example, you should define the **synchronous, dialog-free** method inputBackgroundTest for your object type IDOCTEST. The ABAP coding appears as follows:

```
INCLUDE <OBJECT>.  " Workflow-Makros zur Objektdefinition
begin_data object. " Do not change.. DATA is generated
* only private members may be inserted into structure private
data:
" begin of private,
"   to declare private attributes remove comments and
"   insert private attributes here...
" end of private,
  begin of key,
      idocnumber like edidc-docnum,
  end of key.
end_data object. " Do not change.. DATA is generated
...
BEGIN_METHOD inputBackgroundTest changing container.
```

```
CALL FUNCTION 'IDOC_INPUT_WF_TESTER'
  EXPORTING idocnumber = object-key-idocnumber
  EXCEPTIONS
    OTHERS = 01.
END_METHOD.
```

The transferred parameter is **IDOCNUMBER**, which the object type IDOCTEST has inherited as a key field from the supertype IDOCAPPL.

You can create the function module IDOC_INPUT_WF_TESTER via forward navigation (double click). The function module reads the data from the IDoc and triggers an event if an error occurs (by calling the workflow function module SWE_EVENT_CREATE). In the form routine READ_IDOC_TESTER, you can see how data from an **IDoc table** is turned into **application tables** via **segment structures**.

In the following coding it is often referred to the analog for direct inbound processing (for example, IDOC_INPUT_TESTER), however only for reasons of space, since you must select **either** the direct **or** the Workflow Inbound Processing.

Administration parameters for IDOC_INPUT_WF_TESTER:

Application abbreviation	**V** (Sales and Distribution)
Processing type	normal, start immediately

Coding example

```
FUNCTION IDOC_INPUT_WF_TESTER.
*---------------------------------------------------------
*Globale Schnittstelle:
* IMPORTING  idocnumber LIKE EDIDC-DOCNUM
*---------------------------------------------------------

INCLUDE <CNTN01>.  " Workflow-Makros zur Containerdefinition
DATA:
i_edids LIKE edi_ds OCCURS 1 WITH HEADER LINE,
event_id like swedumevid-evtid,
status like edids-status,
cidocnumber like sweinstcou-objkey,
errorcode like sy-subrc.
      "Für Test der Ausnahmebehandlung
swc_container       ev_container.

* initialize SET/GET parameters and internal tables
PERFORM INITIALIZE_ORGANIZATIONAL_DATA.

CALL FUNCTION 'EDI_DOCUMENT_OPEN_FOR_PROCESS'
     EXPORTING
             document_number = idocnumber
     IMPORTING
```

Defining Inbound Processing as an Object Method

```
          idoc_control             = idoc_contrl
     EXCEPTIONS
          document_foreign_lock    = 01
          document_not_exist       = 02
          document_number_invalid  = 03
          document_is_already_open = 04.

CALL FUNCTION 'EDI_SEGMENTS_GET_ALL'
     EXPORTING
          document_number         = idocnumber
     TABLES
          idoc_containers         = idoc_data
     EXCEPTIONS
          document_number_invalid = 01
          end_of_document         = 02.

PERFORM READ_IDOC_TESTER.

PERFORM CALL_VA01_IDOC_ORDERS
            using errorcode.

PERFORM WRITE_STATUS_RECORD TABLES i_edids     "Write status
      USING errorcode.

CALL FUNCTION 'EDI_DOCUMENT_CLOSE_PROCESS'
     EXPORTING
          document_number     = idocnumber
     IMPORTING
          idoc_control        = idoc_contrl
     EXCEPTIONS
          document_not_open   = 01
          failure_in_db_write = 02
          parameter_error     = 03
          status_set_missing  = 04.
IF errorcode <> 0.
  cidocnumber = idocnumber.
  CALL FUNCTION 'SWE_EVENT_CREATE'
       EXPORTING
            objtype              = 'IDOCTEST'
            objkey               = cidocnumber
            event                = 'InputErrorOccurred'
            start_recfb_synchron = 'X'               "synchroner Aufruf
       IMPORTING
            event_id             = event_id
       TABLES
            event_container      = ev_container
       EXCEPTIONS
            objtype_not_found    = 1
            others               = 2.
  COMMIT WORK.
ENDIF.
ENDFUNCTION.
```

```
FORM INITIALIZE_ORGANIZATIONAL_DATA
...
* Form routine as in section
* Creating a function module(direct inbound processing) [Page 209]

ENDFORM.                    " INITIALIZE_ORGANIZATIONAL_DATA

FORM READ_IDOC_TESTER.
...
* Form routine and called routines as in section
* Creating a function module(direct inbound processing) [Page 209]

ENDFORM.                    " READ_IDOC_TESTER


FORM CALL_VA01_IDOC_ORDERS using errorcode.

* call transaction first dynpro
  PERFORM DYNPRO_START.
* call transaction double-line entry
  PERFORM DYNPRO_DETAIL2.
* incoterms
  PERFORM DYNPRO_HEAD_300.
* call transaction item datas
  PERFORM DYNPRO_POSITION.
  PERFORM DYNPRO_SET USING 'BDC_OKCODE' 'SICH'.
* call transaction VA01
 CALL TRANSACTION 'VA01' USING    BDCDATA
                         MODE     'N'
                         UPDATE   'S'
                         MESSAGES INTO XBDCMSGCOLL.

errorcode = SY-SUBRC.       " remember returncode for status update

ENDFORM.                    " CALL_VA01_IDOC_ORDERS

FORM DYNPRO_START.
...
* Form routine and called routines as in section
* Creating a function module(direct inbound processing) [Page 209]

ENDFORM.                    " DYNPRO_START


FORM WRITE_STATUS_RECORD TABLES i_edids STRUCTURE edi_ds
USING errorcode.

* fill status record
 I_EDIDS-DOCNUM = IDOC_CONTRL-DOCNUM.
 I_EDIDS-LOGDAT = SY-DATUM.
 IF errorcode = 0.
   I_EDIDS-STATUS = BELEG_GEBUCHT.                "value 53
   GET PARAMETER ID 'AUN' FIELD BELEGNUMMER.
```

Defining Inbound Processing as an Object Method

```
   I_EDIDS-STAMID = 'V1'.
   I_EDIDS-STAMNO = '311'.
   I_EDIDS-STAPA1 = 'Terminauftrag'.
   I_EDIDS-STAPA2 = BELEGNUMMER.
 ELSE.
I_EDIDS-STATUS = BELEG_NICHT_GEBUCHT.              "value 51
    I_EDIDS-STAMID = SY-MSGID.
    I_EDIDS-STAMNO = SY-MSGNO.
    I_EDIDS-STAPA1 = SY-MSGV1.
    I_EDIDS-STAPA2 = SY-MSGV2.
    I_EDIDS-STAPA3 = SY-MSGV3.
    I_EDIDS-STAPA4 = SY-MSGV4.
  ENDIF.

CALL FUNCTION 'EDI_DOCUMENT_STATUS_SET'
EXPORTING
          document_number         = idoc_contrl-docnum
          idoc_status             = i_edids
     IMPORTING
          idoc_control            = idoc_contrl
     EXCEPTIONS
          document_number_invalid = 01
          other_fields_invalid    = 02
          status_invalid          = 03.
ENDFORM.                                " WRITE_STATUS_RECORD

Global data of IDOC_INPUT_WF_TESTER
data:
idoc_data LIKE edidd occurs 10 WITH HEADER LINE,
idoc_contrl LIKE edidc occurs 1 WITH HEADER LINE.
* Further data like in section
* Creating a function module (direct inbound processing) [Page 209]
```

Creating a Task (Inbound Processing Using Workflow)

In this step, you create a task for inbound processing which is processed in the background. The task refers to your new object method. Alternatively, you can define a workflow.

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189]

Creating a Task

1. Choose *Development* → *Business Workflow* → *Definition tools* → *Tasks* → *Create* (task type *standard task*). Enter a task abbreviation.

   Choose  and save in the connection.

   When saving assign the task to the same development class as your other development objects (basic type, function group and so on).

2. Assign possible agents to the task: Choose *Additional data* → *Agent assignment* → *Maintain*. You must therefore only perform this step if you do not want to create a background task - in this case there is no agent.

   You can maintain the task as a general task by selecting *Attributes.*

3. Enter the object type and method and set an indicator for the task according to the processing type (for example for *background processing*). Change the *release status* to *implemented*. Save your entries.

   You can check whether you have been entered as a possible agent in the dialog tasks. Choose *Development* → *Business Workflow* → *Runtime tools* → *Start workflow (test environment)* and then enter your task and choose  .

   Make the following entries for the example:

Task abbreviation	**TEST_INPUT**
Method	InputBackgroundTest
Object type	**IDOCTEST**
Execution	Background processing
Agent assignment	none (there are no agents for background tasks)

Creating a Process Code (Inbound Processing via Workflow)

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

1. From the Initial node of the IDoc interface [Ext.] choose *Control* → *Process code inbound*, Navigation *Process code inbound*, and then *New Entries*.
2. Enter a process code, a description and your new standard task in the *Identification* field . Choose the options *processing by task* and, for example, *Processing without ALE* and save your entries.
3. Choose the navigation *Logical Message.*
4. Choose *New entries.*
5. Enter the message type and the process code and save your entries.

   The value help (F4) in the partner profiles can now provide your process code from your message type, that is to say the business view. You can also restrict this help on the screen to specific message variants or functions.

Result

This assignment has specified a new process code for the function module. You can now enter this process code in the partner profiles.

Make the following entries for the example:

Process code	**TESTWFINB**
ID	**<your task>**
Logical message	**TESTER**

Creating a Task (Exception)

In this step, you create a task as the receiver of the event inputErrorOccurred. In addition, you must also activate the type linkage [Ext.].

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

Creating a task by copying

8. From the initial R/3 screen choose *Tools* → *Business Workflow* → *Development, Definition tools* → *Tasks* → *Copy*. Enter *Standard task* as the task type and the ID **00008046** (inbound processing error in ORDERS01).
9. Choose  and enter a task abbreviation. When copying, select the development class which you have configured for yourself in IDoc administration or which applies to all other objects in your basic type definition.

Changing a new task

10. Return to the workflow development menu and choose *Definition tools* → *Task* → *Change*. The system transfers your new task to the input field.
11. Choose  . Replace the object type IDOCORDERS with your new object type for the following tab pages:
    - *Basic data*
    - *Triggering Events* (add new rows, delete old rows)
    - *Terminating Events* (add new rows, delete old rows)

    The object methods *inputForeground* for exception handling are copied from object type IDOCORDERS (tab page *basic data*).
12. Activate the type linkage for *triggering event* (green light) by clicking the button. Save your entries.
13. Select *Additional data* → *Agent assignment* → *Maintain,* to assign possible agents to your task. Via *Attributes* you can classify the task as a general task, that is to say, every user is a possible agent.
14. Save your entries.

You can check whether you have been entered as a possible agent for the task. From the workflow development menu choose *Runtime tools* → *Start workflow (test environment)* and then enter your entries and choose  . If an error message appears which indicates that you are not a possible agent, this may be due to the internal buffer tables used in organization modeling. Then choose *Refresh Organization Environment* and try again.

Creating a Task (Exception)

Make the following entries for the example:

Task abbreviation	**TEST_ERROR**
Object type	**IDOCTEST**
Object method	**errorProcess**
Agent assignment	*general task*

Defining a Partner (Inbound Processing via Workflow)

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] .

Procedure

The new basic type must be entered in the partner profiles before being used. In inbound processing, define the new process code for inbound processing and exception handling for the basic type.

For more information about the procedure, see: Partner Profiles in the Standard Dialog [Page 74]

In the example, you should maintain the following fields in the inbound partner profiles.

Field	Value
Partner number	**TESTCUST**
Partner function	**SP**
Partner type	**KU**
Message type	**TESTER**
IDoc type	**TESTER01**
Permitted agent	Your SAP user, type **US** (individual user), language **EN**
Process code	**TESTWFINB**
Processing	*Immediate processing*

Testing Inbound Processing

Prerequisites

You must have completed the required steps for inbound processing in:

- Extending an IDoc Type [Page 172] (if you have extended a basic type) **or**
- Defining and Using a New Basic Type [Page 189] (if you have defined a new basic type)

Procedure

7. You should use the inbound test tool [Page 96] , as IDocs of the relevant type can be edited directly. Select the following as test templates:
   - An outbound IDoc, if you have configured and tested outbound processing for your IDoc type. You must then exchange the sender and recipient entries and enter 2 as the direction in the control record.
   - Your new IDoc type, if you do not yet have any IDocs of the new type (extended or new basic type). In this case, you must also maintain the segment fields with application data.

   In direct ALE inbound processing, you can access the function module directly and in debugging mode from the test tool. This can be very helpful if new developments are involved. In inbound processing via workflow test your processing, for example, in the ABAP Workbench.
8. Check the current status of the new IDoc using the IDoc display [Page 117] function. If the application has accepted the document, the **current** IDoc status is set to 53 (“application document posted”). Otherwise, the current status is 51 (“application document not posted”). You can display the errors which have been written to the status records by the application by clicking on the records.
9. Check whether the application has written the IDoc data to the correct tables.

   If the test tool displays an error message (for example, “IDoc could not or should not be processed”), you can jump from this message to the IDoc display directly.

   In the example, you should make the following entries in the control record:

Recipient:	Port SAPC11, partner number C11CLNT<current client>, partner type LS (logical system), leave the partner function blank
Sender:	Port TESTPORT, partner number TESTCUST, partner type KU (customer), partner function AG (SP)

If you have extended the basic type ORDERS01:

IDoc type	ORDERS01
Extension	Z1ORDERS

Message type	ORDERS

If you have defined the new basic type TESTER01:

IDoc type	TESTER01
Message type	TESTER

Testing Exception Handling

You can test exception handling by using the test tool to deliberately cause an error, for example sending an IDoc with an empty mandatory field.

Prerequisites

You must have completed the required steps in Defining and Using a Basic Type [Page 189] . You must ensure that your coding calls exception handling (or triggers exception handling in ALE inbound processing via the workflow return parameters) and sets the status value 51.

Procedure

6. From the initial node of the IDoc interface [Ext.] choose *Test* → *Test tool*. Choose *Existing IDoc* as the template for the test and enter the number of an inbound IDoc, which has status 53.
7. Choose  .
8. Use the mouse to click on one of the mandatory segments and delete the contents of the segment.
9. Choose *Standard inbox*. Your IDoc should be assigned error status 51 (you can check using the IDoc display [Page 117] function).
10. If you are entered as a possible agent for your standard task and as a permitted agent in the inbound partner profiles, a work item is sent to your integrated inbox and can be executed from there. Your inbox can be accessed from the IDoc interface initial node by selecting *IDoc* → *Inbox*.

    In the example, you should delete the value from the segment field E1HEAD-VKORG. As a result, an error occurs during inbound processing in the form routine CALL_VA01_IDOC_ORDERS (no sales organization entered). As a possible and selected agent, a work item is sent to your integrated inbox.

Releasing New Objects

Prerequisites

You must have completed the required steps in:

- Extending an IDoc Type [Page 172] (if you have extended a basic type) **or**
- Defining and Using a New Basic Type [Page 189] (if you have defined a new basic type)

Procedure

You should release the new segments first, then your basic type or extension.

You can set the release from the relevant development tool (IDoc type editor or segment editor) by selecting *Edit → Set release.* The release can also be reset by selecting *Edit → Cancel release.*

In the example, you should release the following development objects in the specified order when testing has been completed successfully:

If you have extended the basic type ORDERS01:

Segment	Z1TEST1
Extension	Z1ORDERS

If you have defined the new basic type TESTER01:

Segments	E1HEAD, E1ITEM
IDoc type	TESTER01

Result

Your new definition is now closed. If you still want to translate the documentation into your new definition, you can do this specifically with the general translation transaction. The following section deals with the objects that you need within the translation transaction:

Documentation translation [Page 151]

Troubleshooting in Workflow Processing

Check the following when errors occur in event-driven workflows :

Is an event generated which starts the task?

1. Activate the event monitor: *Tools* → *Business Workflow* → *Development*, *Utilities* → *Events* → *Event Trace* → *On/Off*.
2. Test the inbound or exception handling as described in the checklist.
3. Choose *Tools* → *Business Workflow* → *Development* → *Utilities* → *Events* → *Event Trace* → *Display*. Enter your generated object type and the event and limit the generation time correspondingly.
4. If no event is found, you must check:

   For direct inbound processing via ALE (exception handling):

   - If the function module returns the correct workflow parameter?
   - If the event is maintained for the corresponding object type in the corresponding ALE table (which can be reached via the inbound process code)?

     You can use the ALE consistency check: *Tools* → *Business Framework* → *ALE* → *Development, IDoc Interface* → *Inbound processing* → *Consistency check* and refer to the messages for your process code.

   For inbound processing via workflow:

   - Does your ABAP code for the processed method call the function module SWE_EVENT_CREATE with the correct event?
5. Reactivate the event monitor after this test.

Does the event have a task as event receiver?

1. Use the event monitor as above. If the message *no event receiver* appears in the event list, check whether your task has the correct event in the file card *triggering event.*

IDoc Administration: User Parameters

Use

The IDoc administration function is used to configure **global** and **user specific parameters**. This section describes the user specific or user parameters. For more information about global parameters see: IDoc Administration in Customizing [Page 249]

Features

User parameters for example are:

- the *Test* port for the file interface. For example the paths for Test inbound modified outbound file [Page 101] are derived from this description.
- the output format for the IDoc *documentation*, as well as directory and name for the files outputted (for the HTML/XML/C-header).
- whether you want to automatically start a browser when there is an HTML output of the IDoc *Documentation*
- If you wish to display zero values in the *IDoc lists*, that is, if every time step in the case of IDoc inbound processing is to be displayed, even if no IDocs have been received.

The IDoc administration also provides access

- to *workflow customizing*: Customizing settings can be checked (for example, does a plan variant exist? Is there a workflow administrator?).
- To IDoc CATT test run [Page 103]: Here you can automatically test the interface functions.

Activities

To call the administration function from the Initial node of the IDoc interface [Ext.] choose *Control* → *Administration*. Choose the corresponding tab page to enter default settings or default values.

Additional Settings

Use

The following functions provide default settings which are rarely used.

Features

IDoc Administration in Customizing

Use

The *global parameters* are important IDoc administration default values. They are set in Customizing.

Features

System parameters for example are:

- the *IDoc administrator*, which is notified in exception handling. If necessary it is overwritten with special entries in the partner profiles (see Role Resolution in Exception Handling [Page 30]).
- the *system environment*: It describes the functions with which the IDoc interface can operate: the applications or message control.
- The *maximum number of logged errors* before exception handling is started after the syntax check. You can assign the syntax check in the partner profiles of Outbound processing [Page 76] or Inbound processing [Page 82] .
- the SAP*office inbox folder* for IDocs from the Internet (MIME attachment). IDocs are processed further from this inbox folder when the Internet port type is used. For details on the settings still to be made for this port type read Port type Internet [Page 59].
- Whether errors when importing status files should lead to work items (*"warnings*").

Activities

The *global parameters* can be displayed or maintained from the Initial node of the IDoc interface [Ext.] via *Control* → *IDoc Administration* (). Choose the tab page *Global parameter* and if necessary  .

Forward Inbound

Use

Specify a job as a *logical address* for forwarding the data from an inbound IDoc or from a status confirmation.

Integration

At present, inbound forwarding is only used for internal invoices (application V3 = billing, SD module - *Sales and Distribution*).

Activities

Maintain the addresses from the initial screen of the IDoc interface by selecting *Control* → *Forward inbound*.

The logical address must correspond to the address in the IDoc control record (field SNDLAD = logical address of sender or field RCVLAD = logical address of recipient).

Processing in internal clearing is identified via Outbound process code [Page 70] SD08. In processing, an inbound IDoc is created from the outbound IDoc which is forwarded to the logical address.

Generating File Names

Use

The function modules which generate a file name when IDocs are exchanged via a **“file” port** are stored here. These function modules are displayed as a list of possible entries (F4) when you create a “file” port. The standard system already contains some examples of such function modules. You can store your own function modules here.

Activities

You can maintain the function modules from the initial screen of the IDoc interface by selecting *Control → Generate file names.*

The function module must return directory and file as a character string. It can transfer the directory from the port definition (as the standard modules do), but does not have to.

Checking Partners by Partner Type

Use

While you are editing the partner profiles, the system checks whether master data exists for the specified partner or whether the specified partner is of one of the following types

- R/3 User (partner type US)
- Bank
- Logical system

The check programs and access programs for the individual partner types are stored in this table. Check routines for all partner types are supplied with the standard system.

Activities

You can maintain the programs from the Initial screen of the IDoc interface [Ext.] via *Control → Partner types.*

Enter the following parameters if you wish to change the default check routines or create new partner types:

- Partner type (for example KU for customer, LI for vendor, US for R/3 user)
- Report (for example RSETESTP)
- Form routine (for example READ_KNA1 - customer master record is read)