ALE - Application Link Enabling

ALE is an R/3 technology for distribution of data between independent R/3 installations. ALE is an application which is built on top of the IDoc engine. It simply adds some structured way to give R/3 a methodical mean to find sender, receiver and triggering events for distribution data.

Make Use of ALE for Your Developments:
- Transfer master data for material, customer, supplier and more to a different client or system with [BALE]
- Copy your settings for the R/3 classification and variant configurator to another system, also in [BALE]
- Copy pricing conditions with ALE from the conditions overview screen (e.g. [V12])
14.1 A Distribution Scenario Based On IDocs

ALE has become very famous in business circles. While it sounds mysterious and like a genial solution, it is simply a mean to automate data exchange between SAP systems. It is mainly meant to distribute data from one SAP system to the next. ALE is a mere enhancement of SAP-EDI and SAP-RFC technology.

Imagine your company has several sister companies in different countries. Each company uses its own local SAP installation. When one company creates master data e.g. material or customer master it is much likely that these data should be known to all associates. ALE allows to immediately trigger an IDoc sent to all associates as soon as the master record is created in one system.

Another common scenario is, that a company uses different installations for company accounting and production and sales. In that case ALE allows you to copy the invoices created in SD immediately to the accounting installation.

ALE defines the logic and the triggering events who describe how and when IDocs are exchanged between the systems.

ALE is an application put upon the IDoc and RFC mechanisms of SAP. ALE defines a set of database entries, which are called the ALE scenario. These tables contain the information which IDocs shall be automatically replicated to one or more connected R/3-compatible data systems.

To be clear: ALE is not a new technology. It is only a handful of customizing settings and background routines that allow timed and triggered distribution of data to and from SAP or RFC-compliant systems. ALE is thus a mere enhancement of SAP-EDI and SAP-RFC technology.

14.2 Example ALE Distribution Scenario

To better understand let us model a small example ALE scenario for distribution of master data between several offices.

Let us assume that we want to distribute three types of master data objects, the material master, the creditor master and the debtor master.

Let us assume that we have four offices. This graphic scenario shows the type of data exchanged between the offices. Any of these offices operates an own stand alone R/3 system. Data is exchanged as IDocs which are sent from the sending office and received from the receiving office.

Illustration 18: ALE distribution scenario

<table>
<thead>
<tr>
<th>Data Object</th>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATMAS</td>
<td>New York Office</td>
<td>Venice Office</td>
</tr>
<tr>
<td>CREMAS</td>
<td>Los Angeles</td>
<td>Venice Office</td>
</tr>
<tr>
<td>DEBMAS</td>
<td>Paris Office</td>
<td>Venice Office</td>
</tr>
<tr>
<td>DEBMAS</td>
<td>Paris Office</td>
<td>Venice Office</td>
</tr>
<tr>
<td>DEBMAS</td>
<td>Paris Office</td>
<td>Venice Office</td>
</tr>
<tr>
<td>DEBMAS</td>
<td>Paris Office</td>
<td>Venice Office</td>
</tr>
<tr>
<td>DEBMAS</td>
<td>Paris Office</td>
<td>Venice Office</td>
</tr>
<tr>
<td>DEBMAS</td>
<td>Paris Office</td>
<td>Venice Office</td>
</tr>
</tbody>
</table>

Illustration 19: Scenario in tabular form
### 14.3 ALE Distribution Scenario

ALE is a simple add-on application propped upon the IDoc concept of SAP R/3. It consists on a couple of predefined ABAPs which rely on the customizable distribution scenario. These scenarios simply define the IDoc types and the pairs of partners which exchange data.

ALE defines the logic and the triggering events which describe how and when IDocs are exchanged between the systems. If the ALE engine has determined which data to distribute, it will call an appropriate routine to create an IDoc. The actual distribution is then performed by the IDoc layer.

**The predefined distribution ABAPs can be used as templates for own development**

ALE is of course not restricted to the data types which are already predefined in the BALE transaction. You can write your ALE distribution handlers, which should only comply with some formal standards, e.g. not bypassing the ALE scenarios.

**ALE uses IDocs to transmit data between systems**

All ALE distribution uses IDocs to replicate the data to the target system. The ALE applications check with the distribution scenario and do nothing more than calling the matching IDoc function module, which is alone responsible for gathering the requested data and bringing them to the required data port. You need to thoroughly understand the IDoc concept of SAP beforehand, in order to understand ALE.

The process is extremely simple: Every time a data object, which is mentioned in an ALE scenario, changes an IDoc is triggered from one of the defined triggering mechanisms. These are usually an ABAP or a technical workflow event.

**ABAPs can be used in batch routine**

Distribution ABAPs are started manually or can be set up as a triggered or timed batch job. Sample ABAPs for ALE distribution are those used for master data distribution in transaction BALE, like the ones behind the transaction BD10, BD12 etc.

**Workflows triggered from change document**

The workflow for ALE is based on change points. Change points are entries in a special database entity, which record the creation or modification of a database object. These change points are very much like the SAP change documents. They are also written from within a change document, i.e. from the function CHANGEDOCUMENT_CLOSE. The workflow is also triggered from within this function.

**Relevance for change points is defined in IMG**

SAP writes those ALE change points to circumvent a major drawback of the change documents. They are also written from within a change document. Change documents are and only are written, if a value of a table column changes, if this column is associated with a data element which is marked as relevant for change documents [see SE11]. ALE change points use a customized table, which contains the names of those table fields, which are relevant for change points.

### 14.4 Useful ALE Transaction Codes

ALE is customized via three main transactions. These are SALE, WEDI and BALE.

This is the core transaction for SALE customizing. Here you find everything ALE related, which is not already covered by the other customizing transactions.

**BALE** - ALE Specific Customizing

WEDI - IDoc

Here you define all the IDoc related parts, which make up most of the work related to ALE.

**BALE** - Central menu

This is a menu, which combines most function necessary for ALE distribution, especially the triggering of manual distribution of master data or variant configuration or classification.

**BALE** - Automatically Generate IDocs From A BAPI

Good stuff for power developers. It allows to generate all IDoc definitions including segments and IDoc types from the DDIC entries for a BAPI definition.
14.5 ALE Customizing (SALE)

ALE customizing is relatively straightforward. The only mandatory task is the definition of the ALE distribution scenario. The other elements did not prove as being very helpful in practical applications.

SALE

All ALE special customizing is done from within the transaction SALE, which links you to a subset of the SAP IMG.

Distribution Scenarios

The scenario defines the IDoc types and the pairs of IDoc partners which participate in the ALE distribution. The distribution scenario is the reference for all ABAPs and functionality to determine, which data is to be replicated and who could be the receiving candidates. This step is of course mandatory.

Change Pointers

The change pointers can be used to trigger the ALE distribution. This is only necessary if you really want to use that mechanism. You can however always send out IDocs every time an application changes data. This does not require the set-up of the change pointers.

Filters

SAP allows the definition of rules, which allow a filtering of data, before they are stored in the IDoc base. This allows you to selective accept or decline individual IDoc segments.

Conversion

ALE allows the definition of conversion rules. These rules allow the transition of individual field data according mapping tables. Unfortunately the use of a function module to convert the data is not realized in the current R/3 release.

Conversion

The filter and conversion functionality is only attractive on a first glance. From practical experience we can state, that they are not really helpful. It takes long time to set up the rules and rules usually are not powerful enough to avoid modifications in an individual scenario. Conversion rules tend to remain stable, after they have once been defined. Thus it is usually easier to call an individual IDoc processing function module, which performs your desired task more flexible and easier.
**14.6 Basic Settings**

Basic settings have to be adjusted before you can start working with ALE.

**Logical System**

Before we start we need to maintain some logical systems. This names for the RFC destinations which are used as communication partners. An entry for the logical system is created in the table TBDLS.

**Illustration 19: SM31 - View Maintenance TBDLS**

Assign logical system to a client

You will finally have to assign a logical system to the clients involved in ALE or IDoc distribution. This is done in table T000 which can be edited via SM31 or via the respective ALE tree element.

**Illustration 20: SM31 - View Maintenance T000**

**14.7 Define The Distribution Model (The "Scenario")**

The distribution model (also referred to as ALE-Scenario) is a more or less graphical approach to define the relationship between the participating senders and receivers.

The distribution model is shared between all participating partners. It can therefore only be maintained in one of the systems which we shall call the leading system. Only one system can be the leading system, but you can set the leading system to any of the partners at any time, even if the scenario is already active.

**BD64**

This will be the name under which you will address the scenario. It serves as a container in which you put all the from-to relations.

**Illustration 21: Create a model view**

Suggestion: One scenario per administration area

You can have many scenarios for eventual different purposes. You may also want to put everything in a single scenario. As a rule of thumb it proved as successful, that you create one scenario per administrator. If you have only one ALE administrator, there is no use of having more than one scenario. If you have several departments with different requirements, that it might be helpful to create one scenario per department.

**Illustration 22: Add a message type to the scenario**
Define the Distribution Model (The "Scenario")

14.8 Generating Partner Profiles

A very useful utility is the automatic generation of partner profiles out of the ALE scenario. Even if you do not use ALE in your installation, it could be only helpful to define the EDI partners as ALE scenario partners and generate the partner profiles.

If you define the first profile for a partner, you have to create the profile header first. Click on the blank paper sheet.

The values given here are not really important. The partner class is only a classification value. You can give an arbitrary name in order to group the type of partners, e.g., EDI for external ones, ALE for internal ones and IBM for connection with IBM OS/390 systems.
Illustration 31: Generation Partner Profiles Form SALE menu

Illustration 32: Automatically created partner profile

There have been two profiles generated. The one is for MATMAS which we explicitly assigned in the distribution scenario. The second one is a mandatory IDoc type with the name SYNCH which is used for RFC control information and synchronisation. This one is only created if it does not yet exist.

Illustration 33: Outbound partner profile after generation

Here is a detail view of the parameters generated. The receiver port is the RFC destination, that had been created for TESTTARGET with SM59. Data goes to table EDP13.
There is a very powerful utility which allows to generate most IDoc and ALE interface objects directly from a BAPI's method interface.

Every time BAPI is executed, the ALE distribution is checked.

I will demonstrate the use with the object KNA1 and method CHANGEFROMDATA. This object is executed every time the data of a customer (table KNA1) is modified, e.g. via transactions XD01 or XD02. This object will automatically trigger a workflow event after its own execution, which can be used for the ALE triggering. BBDB will generate an ALE interface with all IDoc definitions necessary. This ALE introduced can be introduced in a scenario. Hence, every time the customer data is modified, the data is going to be distributed as an IDoc according the ALE scenario setup.

Enter the object and the method.
Now you can specify the required IDoc types and the names of the function module and function group for the processing routines. Note, that the development class (Entwicklungsklasse) and the function group (Funktionsgruppe) need to be in your customer name space, i.e. should begin with Y or Z. The values proposed on this screen are usually inappropriate.

Result report
Click on generated objects to see what was generated in detail.

Illustration 35: Generation protocol
A detailed report is shown. The report is clickable so that you can directly view the generated objects. The hotspot will appear when you move over a clickable object. The transaction has generated an IDoc type.

The IDoc type is generated with a header section containing the interface values of the object, and a data section with the remaining fields of the object data structure. The BAPIs interface definition looks like that.
FUNCTION bapi_customer_changefromdata.

**IMPORTING**

VALUE(PI_ADDRESS) LIKE BAPIKNA101 STRUCTURE BAPIKNA101
VALUE(PI_SALESORG) LIKE BAPIKNA102-SALESORG
VALUE(PI_DISTR_CHAN) LIKE BAPIKNA102-DISTR_CHAN
VALUE(PI_DIVISION) LIKE BAPIKNA102-DIVISION
VALUE(CUSTOMERNO) LIKE BAPIKNA103-CUSTOMER

**EXPORTING**

VALUE(PE_ADDRESS) LIKE BAPIKNA101 STRUCTURE BAPIKNA101
VALUE(RETURN) LIKE BAPIRETURN STRUCTURE BAPIRETURN

Illustration 36: Function interface of the BAPI

Generated segment structure from BAPI function interface parameter

For each of the parameters in the BAPIs interface, the generator created a segment for the IDoc type. Some segments are used for IDoc inbound only, others for IDoc outbound instead. Parameter fields that are not structured will be combined in a single segment which is placed as first segment of the IDoc type and contains all these fields. This collection segment receives the name of the IDoc type. In our example this is the generated segment Z1ZAXX_KNA1_CHANGED.

The segment below has been created as a header level segment and combines all function module parameters, which do not have a structure, i.e. which are single fields. E.g. if the BAPI has parameters a parameter i_material LIKE mara-matnr then it will be placed in the control segment. However if it is declared i_material STRUCTURE mara then it will create an own IDoc segment.

### Illustration 37: Segment Z1ZAXX_KNA1_CHANGED

<table>
<thead>
<tr>
<th>Port</th>
<th>Fieldname</th>
<th>Description</th>
<th>Mandatory</th>
<th>IDOCExp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PI_SALESORG</td>
<td>WERKS</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>2</td>
<td>PI_DISTR_CHAN</td>
<td>VWERE</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>3</td>
<td>PI_DIVISION</td>
<td>SPART</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>4</td>
<td>CUSTOMERNO</td>
<td>KUNNR</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

14.10 Defining Filter Rules

ALE allows to define simple filter and transformation rules. These are table entries, which are processed every time the IDoc is handed over to the port. Depending on the assigned path this happens either on inbound or outbound.

Rules are defined with the `SALE` transaction.

Illustration 38: `SALE`

Illustration 39: Assigning the conversion rule to an IDoc segment
Workflow Technology

There are two faces of workflow in R/3. There is once the business oriented workflow design as it is taught in universities. This is implemented by the SAP Business Workflow™. However, the workflow is also a tool to link transactions easily. It can be used to easily define execution chains of transactions or to trigger user actions without the need to modify the SAP standard code. This can even be achieved without laboriously customizing the HR related workflow settings.

Summary

- Workflow event linkage allows the execution of another program when a transaction finishes.
- The workflow event linkage mechanism can be easily used without customizing the full workflow scenarios.
- This way we use the workflow engine to chain the execution of transaction and circumvent the setup of the SAP Business Workflow™.
- There are several independent ways to trigger the workflow event linkage.

Americans work hard because they are optimists. Germans work hard because they fear the future.
Workflow in R/3 and Its Use For Development

15.1 Workflow in R/3 and Its Use For Development

SAP R/3 provides a mechanism, called Workflow, that allows conditional and unconditional triggering of subsequent transactions from another transaction. This allows to build up automatic processing sequences without having the need to modify the SAP standard transactions.

Workflow as business method

The SAP business workflow was originally designed to model business workflows according to scientific theories with the same name Business Workflow. This is mainly a modelling tool, that uses graphical means, e.g., flow charting, to sketch the flow of events in a system to achieve the required result. SAP allows to transcript these event modelling into customizing entries, which are then executed by the SAP Workflow mechanism.

Transaction SWO1

The transaction to enter the graphical model, to define the events and objects and to develop necessary triggering and processing objects, is SWO1 (it is an O not a zero).

SAP approach unnecessary complex and formal

I will not even try to describe, how to design workflows in SAP. I believe, that the way how workflows are realized in SAP is far to complicated and unnecessarily complex and will fill a separate book.

Workflow events can be used for own developments

Fortunately the underlying mechanism for workflows is less complex as the formal overhead. Most major transactions will trigger the workflow via SW_EVENT_CREATE. This will make a call to a workflow handler routine, whose name can usually be customized dynamically and implemented as a function module.

15.2 Event Coupling (Event Linkage)

Contrary to what you mostly hear about R/3 workflow, it is relatively easy and mechanical to define a function module as a consecutive action after another routine raised a workflow event. This can e.g. be used to call the execution of a transaction after another one has finished.

Every workflow enabled transaction will call SW_EVENT_CREATE

The whole workflow mechanism is based on a very simple principle. Every workflow enabled transaction will call directly or indirectly the function module during SW_EVENT_CREATE update.

SW_EVENT_CREATE will look in a table, e.g. SWTYPECOU to get the name of the following action

The function module SW_EVENT_CREATE will then consult a customizing table. For a simple workflow coupling, the information is found in the table SWTYPECOU. The table will tell the name of the subsequent program to call, either a function module or an object method.

This way of defining the subsequent action is called type coupling because the action depends on the object type of the calling event.

The call to the following event is done with a dynamic function call. This requires, that the called function module has a well-defined interface definition. Here you see the call as it is found in SW_EVENT_CREATE.

Program 10: This is the call of the type coupled event in release 40B

```
CALL FUNCTION typecou-regefb ' call receiver_type_get_fb
EXPORTING
  obj_type = typecou-obj_type
  obj_key = obj_key
  event = event
  generic_rectype = typecou-rectype
IMPORTING
  rectype = typecou-rectype
TABLES
  event_container = event_container
EXCEPTIONS
  OTHERS = 1.
```

reading the change pointers which are not yet processed

REDMIDOC

The ABAP REDMIDOC will process all open change pointers and distribute the matching IDocs.
15.3 Workflow from Change Documents

Every time a change document is written a workflow event for the change document object is triggered. This can be used to chain unconditionally an action from a transaction.

The most interesting chaining point for workflow events is the creation of the change document. Nearly every transaction writes change documents to the database. This document is committed to the database with the function module `CHANGEDOCUMENT_CLOSE`. This function will also trigger a workflow event.

The workflow handler triggered by an event which is fired from change documents is defined in table `SWECDOBJ`. For every change document type a different event handler can be assigned. This is usually a function module and the call for it is the following:

```
CALL FUNCTION 'swecdobj-objtypefb'
EXPORTING
  changedocument_header = changedocument_header
  objecttype = swecdobj-objtype
IMPORTING
  objecttype = swecdobj-objtype
TABLES
  changedocument_position = changedocument_position.
```

Program 11: This is the call of the change doc event in release 40B

In addition, change pointers for ALE are written.

```
CALL FUNCTION 'CHANGEDOCUMENT_CLOSE'
EXPORTING
  changedocument_header = cdhdr
IMPORTING
  objecttype = swecdobj-objtype
TABLES
  changedocument_position = ins_cdpos.
```

Program 12: This is the call of the type coupled event in release 40B

```
CALL FUNCTION 'CHANGE_POINTERS_CREATE'
EXPORTING
  changedocument_header = cdhdr
TABLES
  changedocument_position = ins_cdpos.
```

15.4 Trigger a Workflow from Messaging

The third common way to trigger a workflow is doing it from messaging.

Define a message for condition technique

When the R/3 messaging creates a message and processes it immediately, then it actually triggers a workflow. You can use this to set up conditional workflow triggers, by defining a message with the message finding and link the message to a workflow.

Assign media "W" or "E"

You define the message the usual way for your application as you would do it for defining a message for SAPscript etc. As a processing media you can assign either the type "W" for workflow or "E" for special processing.

The media type "W" for workflow would require defining an object in the object repository. We will only show how you can trigger the workflow with a standard ABAP using the media type "E".

Form routine requires two parameters

You need to assign a program and a form routine to the message in table `TNAPR`. The form routine you specify needs exactly two USING-parameters as in the example below.

```
REPORT ZSNASTWF.
TABLES: NAST.
FORM ENTRY USING RETURN_CODE US_SCREEN.
  * Here you go
  RETURN_CODE = 0.
  SY-MSGID = '43'.
  SY-MSGNO = '000'.
  SY-MSGV1 = 'Workflow called via NAST'.
  CALL FUNCTION 'NAST_PROTOCOL_UPDATE'
    EXPORTING
      MSG_ARBGB = SYST-MSGID
      MSG_NR    = SYST-MSGNO
      MSG_TY    = SYST-MSGTY
      MSG_V1    = SYST-MSGV1
      MSG_V2    = SYST-MSGV2
      MSG_V3    = SYST-MSGV3
    EXCEPTIONS
      OTHERS    = 1.
  ENDFORM.
```

NAST must be declared public in the called program

In addition, you need to declare the table `NAST` with a tables statement public in the ABAP where the form routine resides. When the form is called the variable `NAST` is filled with the values of the calling message.
15.5 Example, How To Create A Sample Workflow Handler

Let us show you a function module which is suitable to serve as a function module and define the linkage.

Create a function module that will be triggered by a workflow event

We want to create a very simple function module that will be triggered upon a workflow event. This function is called from within function SWE_EVENT_CREATE. The parameters must comply the calling standard as shown below.

```plaintext
CALL FUNCTION typecou-recegetfb
  EXPORTING
    objtype = typecou-objtype
    objkey = objkey
    event = event
    generic_rectype = typecou-rectype
  IMPORTING
    rectype = typecou-rectype
  TABLES
    event_container = event_container
  EXCEPTIONS
    OTHERS = 1.
```

This is the call of the type coupled event in release 40B

Template for workflow handler

Release 40B provides the function module WF_EQUI_CHANGE_AFTER_ASSET which could be used as template for the interface. So we will copy it and put our coding in instead.

```plaintext
FUNCTION Z_WORKFLOW_HANDLER.
  """Lokale Schnittstelle:
  "" ""IMPORTING
    VALUE(OBJKEY) LIKE  SWEINSTCOU-OBJKEY
    VALUE(EVENT) LIKE  SWETYPECOU-EVENT
    VALUE(RECTYPE) LIKE  SWETYPECOU-RECTYPE
    VALUE(OBJTYPE) LIKE  SWETYPECOU-OBJTYPE
  "" ""TABLES
    EVENT_CONTAINER STRUCTURE  SWCONT
  "" ""EXCEPTIONS
    NO_WORKFLOW RECEIVERS-EXPRESS  = ' '.
    RECEIVERS-RECEIVER = SY-SUBRC.
    APPEND RECEIVERS.
    DOCUMENT_DATA-OBJ_DESCR = OBJ_KEY.
    CONTENT = OBJ_KEY.
    APPEND CONTENT.
    CALL FUNCTION 'SO_NEW_DOCUMENT_SEND_API1'
      EXPORTING
        DOCUMENT_DATA              = DOCUMENT_DATA
        OBJECT_CONTENT             = CONTENT
        RECEIVERS                  = RECEIVERS.
  "" ""ENDFUNCTION.
```

Program 13: A workflow handler that sends an Sap Office mail

Link handler to caller

The function can be registered as a handler for an event. This is done with transaction SWLD.

Event logging

If you do not know the object type, that will trigger the event, you can use the event log. You have to activate it from SWLD and then execute the event firing transaction. When the event has been fired it will a trace in the event log.

Hit list of common errors

- You forgot to set the RFC flag in the interface definition of your event handling function module
- There is a syntax error in your function module (check with generate function group)
- You mistyped something when defining the coupling
- The internal workflow destination WORKFLOW_LOCAL_000 is not defined

If you think your handler did not execute at all, you can check the list of pending background tasks with transaction SM58. If your event is not there it has either neither been triggered (so your tables SWETYPEENA and SSWETYPEOBJ may have the wrong entries) or your event handler executed indeed and may probably have done something else than you expected. Ergo: your mistake.

Read carefully the help for CALL FUNCTION .. IN BACKGROUND TASK

Your event handler function is called IN BACKGROUND TASK. You may want to read carefully the help on this topic in the SAP help. (help for “call function” from the editor command line)
Example, How To Create A Sample Workflow Handler

Workflow Technology

Example, How To Create A Sample Workflow Handler

Program 14: Send a SAPoffice mail triggered by a workflow event (full example)