Understanding and Optimizing Your ALE Data Distribution: Minimizing Data Volume

Arthur Wirthensohn

Arthur Wirthensohn is a senior consultant at EDS Switzerland and a member of EDS's international Technical Leadership Network. Currently, he is the project manager and technical lead of the Enterprise Application Integration (EAI) department, which delivers SAP-related services such as ALE/EDI, SAP Business Workflow, Web Application Server, Data Migration, and ABAP Programming Services.

(complete bio appears on page 104)

If you are involved in setting up or maintaining data distribution for a business process using SAP's Application Link Enabling (ALE), you have a role to play in ensuring that the distribution performs well and uses system resources efficiently. Whether you are a network or system administrator, a programmer, an application integration specialist, or a business process consultant, a thorough understanding of how ALE works will enable you to contribute to your team's efforts to optimize your distributed business process. This is the second installment of a two-part article series designed to provide you with that knowledge.

Though individual ALE distributed business processes vary widely, each with its own combination of business scenario, business process, and technical environment, there are two factors that affect the performance of all ALE data distributions — the capacity and availability of processing resources, and the volume of data to be distributed.

The first installment of this article series, published in the November/December 2003 *SAP Professional Journal*, dealt with issues relating to processing capacity and availability. Optimizing distributions at this level requires a deep understanding of how ALE data distributions work — how they utilize work processes, how processing loads can be distributed, when and how to couple or decouple ALE processes, and when and how to use packets for processing. In that article, system administrators learned how to set up the system in a way that provides ALE with enough system resources without negatively affecting other system users. Application integration specialists and business process consultants learned how to design and customize the process properly.

This second installment shows you how to optimize your data distribution by minimizing the amount of data to be processed. An

SAP standard distributed business process — without any filters — usually creates much more information than is actually needed on the receiver side of the distribution. For example, a purchase order might contain a lot of text that is important for the purchaser, but not relevant for the supplier. In a standard message, this unnecessary text might be forwarded to the receiver system. System resources are consumed as the text is created, transmitted, and posted. Minimizing the data volume so that only the necessary data is transmitted is a key way to optimize an ALE data distribution.

If you are a programmer, an application integration specialist, or a business process consultant, you can use what you learn here to reduce the system load of a distributed business process so that it consumes as few system resources as necessary every step of the way, from the sender system to the receiver system. First, you'll learn how to analyze the data to discover what data is required and what is not. Then, you'll see exactly where in the distribution process you can minimize the data using ALE tools and features. Finally, you'll get a detailed explanation, with examples, of how to implement the tools and features that yield the biggest performance gains.

A prerequisite for understanding this article is a basic knowledge of ALE and SAP Business Workflow. Previously published articles in *SAP Professional Journal* are a good resource for that knowledge.[1]

Data Analysis — Identifying Essential, Nonessential, and "Nice-to-Have" Data

There are two basic ways that you can reduce the data volume of an ALE distribution. You can minimize:

- The data of a single message
- The number of messages to be distributed

In either case, you need to determine what data is essential for the business process on the receiver side. Then you can eliminate the nonessential data from the distribution — a key step in reducing data volume. Let's look at how to analyze each scenario so that you can make this determination.

Analyzing the Data of a Single Message

Not every record and field in a message (IDoc or XML) will be required by the target business object[2] of the distribution process. If a sales order is being created in the receiver system, for example, then only the data necessary to create the sales order is required. You will need to analyze the records and fields of the message to see what is needed and what is not. This analysis is necessary whether you have a standard SAP ALE distribution or a custom distribution, because the requirements of each business process are almost always different. For example, in one scenario the order reason field in a sales order might be mandatory, while in another scenario the field is not used.

Be aware that the IDoc structure of a business object does not always correspond to the data structure of the business object. For example, a very common business process is to create an ORDERS IDoc from a purchase order on the sender system and then post that ORDERS IDoc as a sales order on the receiver system. If you compare the structure elements of two different types of orders (e.g., a sales order and a purchase order) and the corresponding IDoc, however, you will see that the structure of the purchase order (**Figure 1**) is different from the structure of the ORDERS IDoc (**Figure 2**), which in turn

[1] "Real-Time, Outbound Interfaces to Non-R/3 Systems Made Simple with Change Pointers, Message Control, and Workflow" (Premiere Issue); "Data Transformation in SAP Standard ALE Distributed Business Processes: How to Ensure an Efficient, Effective Implementation" (July/August 2003); and "Understanding and Optimizing Your ALE Data Distribution: Controlling ALE Processing" (November/December 2003).

[2] The term *business object* is used here in the sense of an object type in SAP's Business Object Repository. Material master, G/L account, and customer sales order are examples of business objects. An instance of a business object is the data representation of the real-world object identified by the key of the object type. For example, customer number 1000 might be an instance of a business object.

Figure 1 Main Structure Elements of a Purchase Order

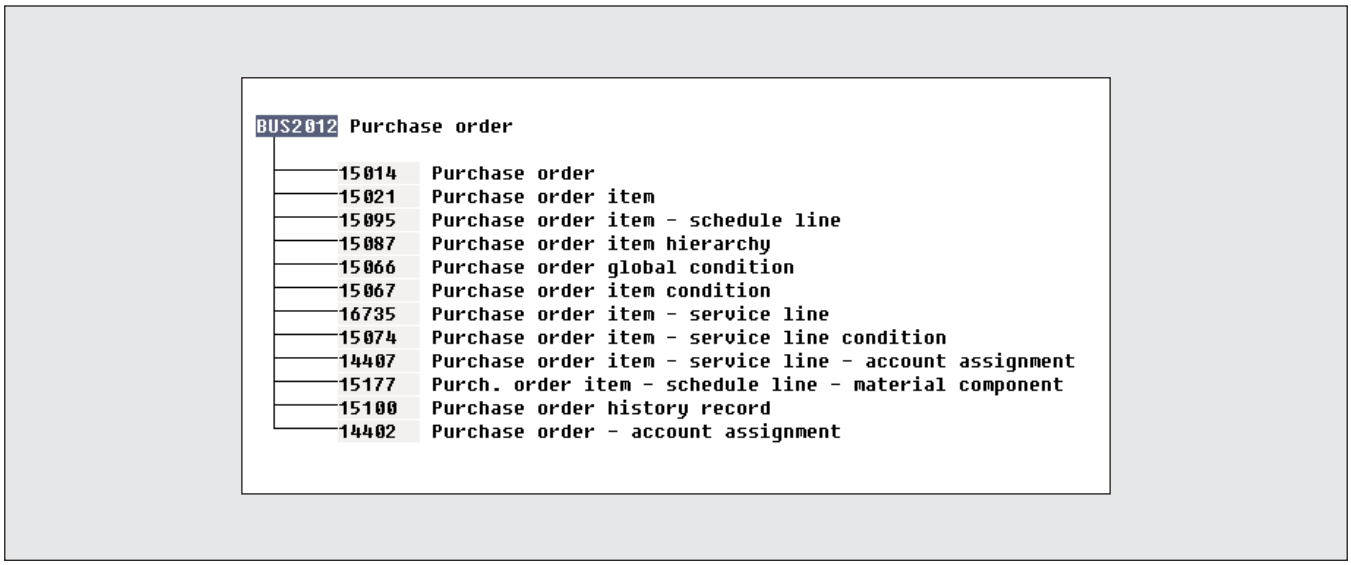


Figure 2 Main Portion of an ORDERS IDoc Structure

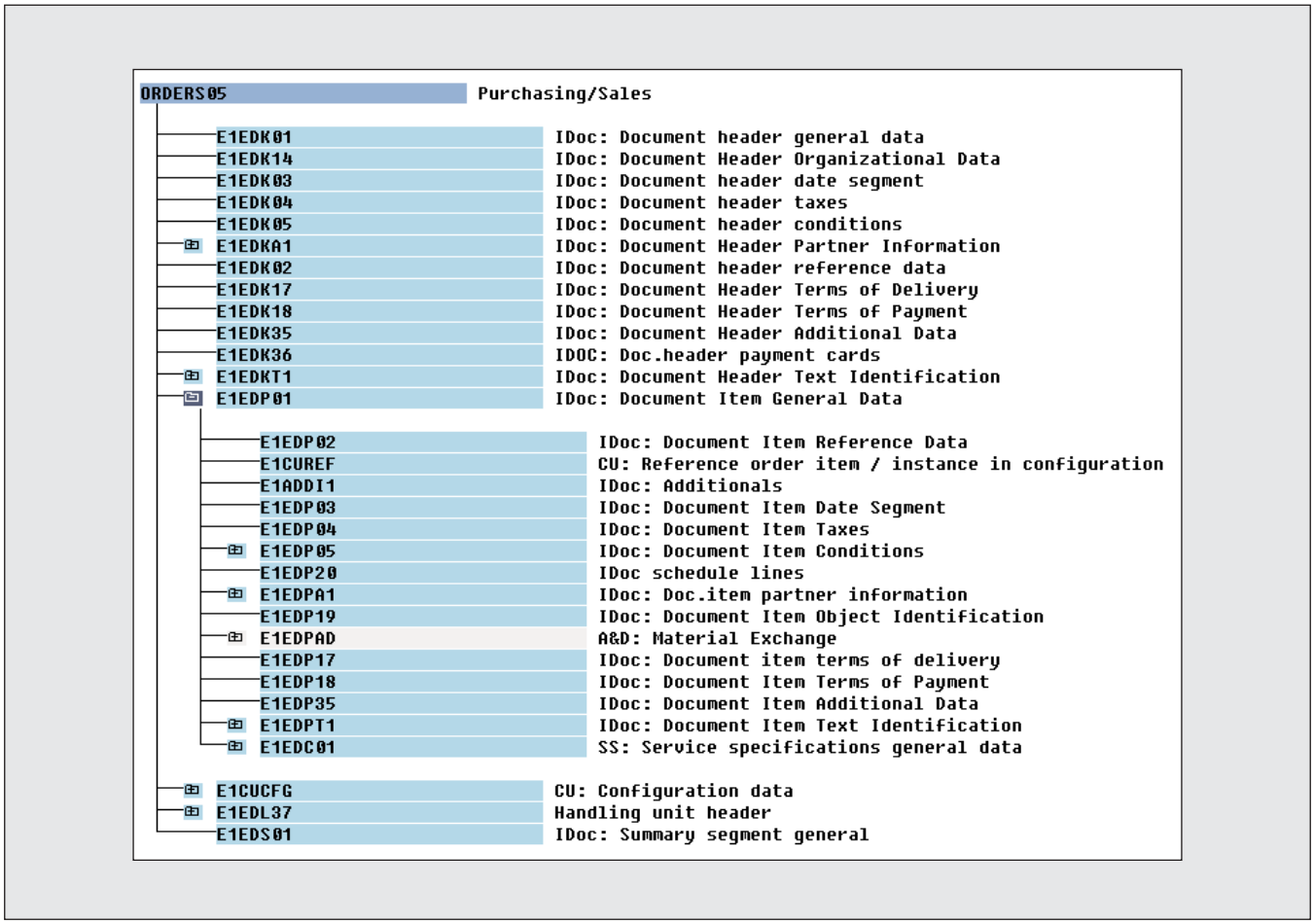
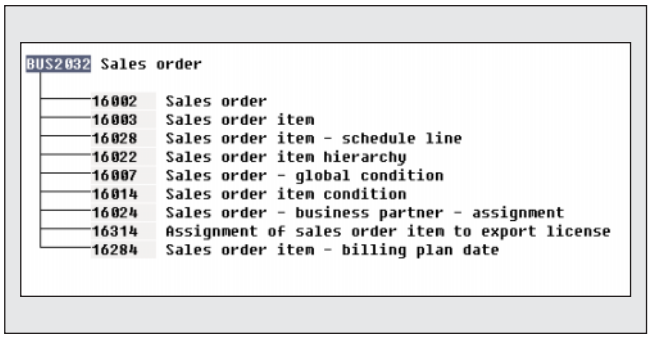


Figure 3 Main Structure Elements of a Sales Order



is different from the structure of the sales order (**Figure 3**).[3]

> ✓ ***Tip***
>
> *You can navigate through the structure of an ORDERS IDoc by calling transaction WE30 or WE60, entering ORDERS05 as the basic type, and pressing the Display button.*

If the IDoc structure of a message does not correspond to the business object, you will first have to determine what data is needed for the business process (for example, what is required to create the sales order) and then determine what elements of the IDoc structure refer to the necessary data elements. Refer to the IDoc documentation in transaction *WE60* for a good explanation of the content of the IDoc's segments and fields. This information will help you to discover the relationship of the IDoc fields to the business document data.

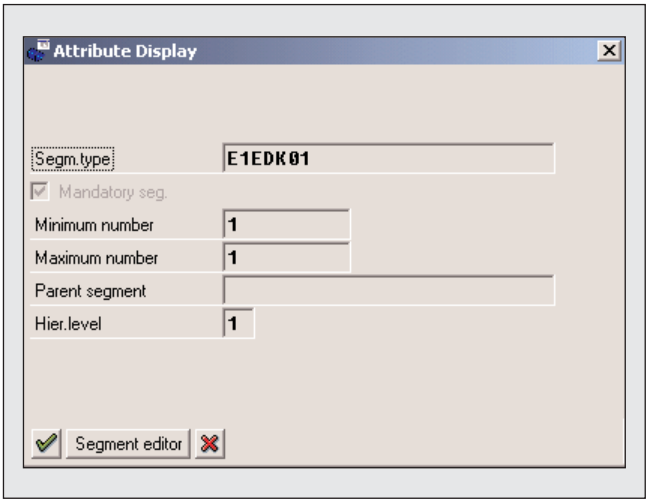
[3] Due to space constraints, I have shown a reduced view of the structure elements of the purchase order and the sales order. If you want to see the entire structure of the business object graphically, go to transaction *SWO1* (Business Object Builder), enter *BUS2012* (purchase order) or *BUS2032* (sales order), and press the *Display* button. On the following screen, press the *BO data model* button and navigate to a description of the business object. On the description screen, press the *Graphic* button (or *Ctrl+F3*) to get a graphical overview of the structure of the business object.

> ✓ ***Note!***
>
> *In an ALE data distribution, you must send, at a minimum, the mandatory segments of an IDoc.*

You can learn what segments of an IDoc are mandatory by using transaction *WE60* (Documentation of IDoc Types) or transaction *WE30* (Develop IDoc Types):

- **To use transaction *WE60*:** Enter the IDoc you are interested in, press the *F8* button, and expand the tree structure. Through SAP Release 4.6C (or Basis Release 4.6D), you will get the tree structure of the IDoc in list form; as of SAP Web Application Server (Web AS) 6.10, you will get an HTML view of the IDoc. Regardless of how the tree is displayed, a field in the header description of the segment indicates whether the segment is mandatory or optional.
- **To use transaction *WE30*:** Enter the IDoc you are interested in, press the *Display* button (*F7*), and expand the tree structure. Double-click on the segment name (or mark the segment and press *F2*). If the *Mandatory seg*. attribute is flagged on the pop-up dialog box (**Figure 4**), the segment is mandatory; otherwise, it is optional.

Figure 4 Properties of an IDoc Segment



Analyzing the Number of Messages to Be Distributed

To minimize the number of messages (IDoc or XML) to be sent, you have to determine which instances of the business object you really need. Then you can define the appropriate selection criteria by analyzing the attributes common to the necessary instances.

Take, for example, a factory that produces goods sold by independent sales companies. The factory and the sales companies all have their own SAP systems, which interface via ALE. The factory supplies the sales companies with material master data to aid them in placing their orders. In distributing this data, the factory can choose to send each company only the data that is relevant to that company by selecting material masters according to sales organization. In this scenario, then, the common attribute for the material masters data would be the sales organization.

What to Do About "Nice-to-Have" Data

When your analysis reveals some necessary data and some clearly unnecessary data, your course is clear: you can simply eliminate the unnecessary data from your ALE distribution. But what if your analysis has also revealed some data that, while not absolutely essential, would be nice to have on the receiver side? How do you decide whether or not to distribute that data? You'll need to know what it costs in terms of system capacity before you can make this decision. I recommend the following strategy for deciding whether or not to distribute this data:

1. Analyze the quantity of the data by counting the corresponding segments in some existing IDocs that carry the entire standard message.

2. Estimate how many of these IDocs will be transmitted in a given period.

3. If the data benefits the business process enough to justify the number of segments to be sent, then by all means include those segments in your minimized set of message data. Otherwise, leave them out.

For example, the factory in the previous example might consider transmitting the forecasting and consumption data of the material masters to the sales companies, because it would be nice for the sales companies to get an idea of the factory's production trends. Analysis of the segments of existing material master IDocs reveals that there are about 500 records (segments) with forecasting and consumption data per IDoc and about 10 records (segments) with the material master data that are really required by the sales company. There will be approximately 10,000 material master IDocs in an initial data transfer to each sales company and about 100 IDocs with data changes transferred each day. In this scenario, the "nice-to-have" forecasting and consumption data would have a serious performance and capacity impact on the interface — 5,000,000 more records in the initial transfer, and about 50,000 more each day. The factory and sales companies would be wise to find another way to get the forecasting and consumption information to the sales companies.

Proof of Concept

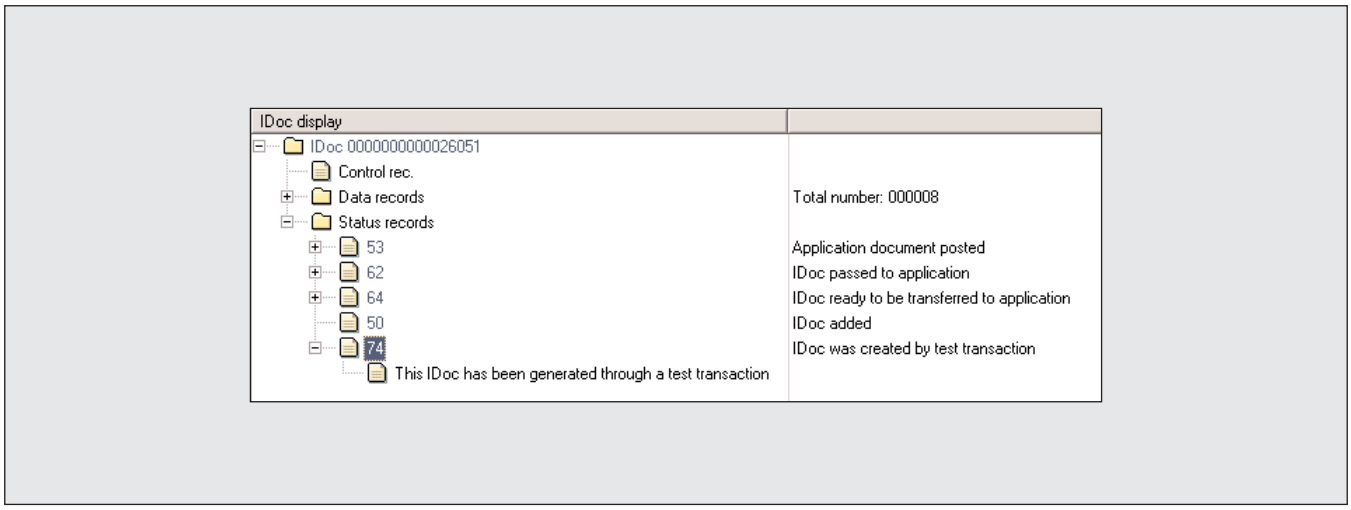
You've analyzed the data and ascertained what must be sent at a minimum. Now you need to verify that an IDoc with the pared-down data set you have defined is able to create the desired new instance of the business object on the receiver system. A helpful tool for conducting your proof of concept is the test tool for processing IDocs (transaction *WE19*). Here are the steps:

1. Call the test tool (on the receiver system, of course) and, in the select-option *Existing IDoc*, choose an IDoc that has already successfully created an instance of the business object in question. This IDoc will serve as your reference for the test.

2. On the following screen, the test tool provides the opportunity to add, change, and delete segments and fields of the reference IDoc. Delete all the information that, according to your analysis, is not required.

3. Process the IDoc using standard inbound

Figure 5 Status Record 74 Is Added After the IDoc Is Processed



processing (*F8*) and check the result in a monitoring transaction (e.g., *WE02*). If you look at the IDoc's status records, you'll see that the test tool has added status 74 (IDoc was created by test transaction) right after status 50 (IDoc added) to indicate its origin (**Figure 5**).

If the posting was successful, check the result in the corresponding business transaction as well (e.g., *VA03* if you tried to create a sales order).

The manipulated IDoc that you have just processed is stored to the IDoc database. Therefore, you can reuse it as a reference if you have to run another test cycle.

> ✓ ***Tip***
>
> *The test tool for processing IDocs provides an IDoc syntax check (F6). After deleting data on the referenced IDoc, you can use the syntax check to make sure that the IDoc still has the appropriate structure. For example, the syntax check will issue an error message if you have deleted a mandatory segment.*

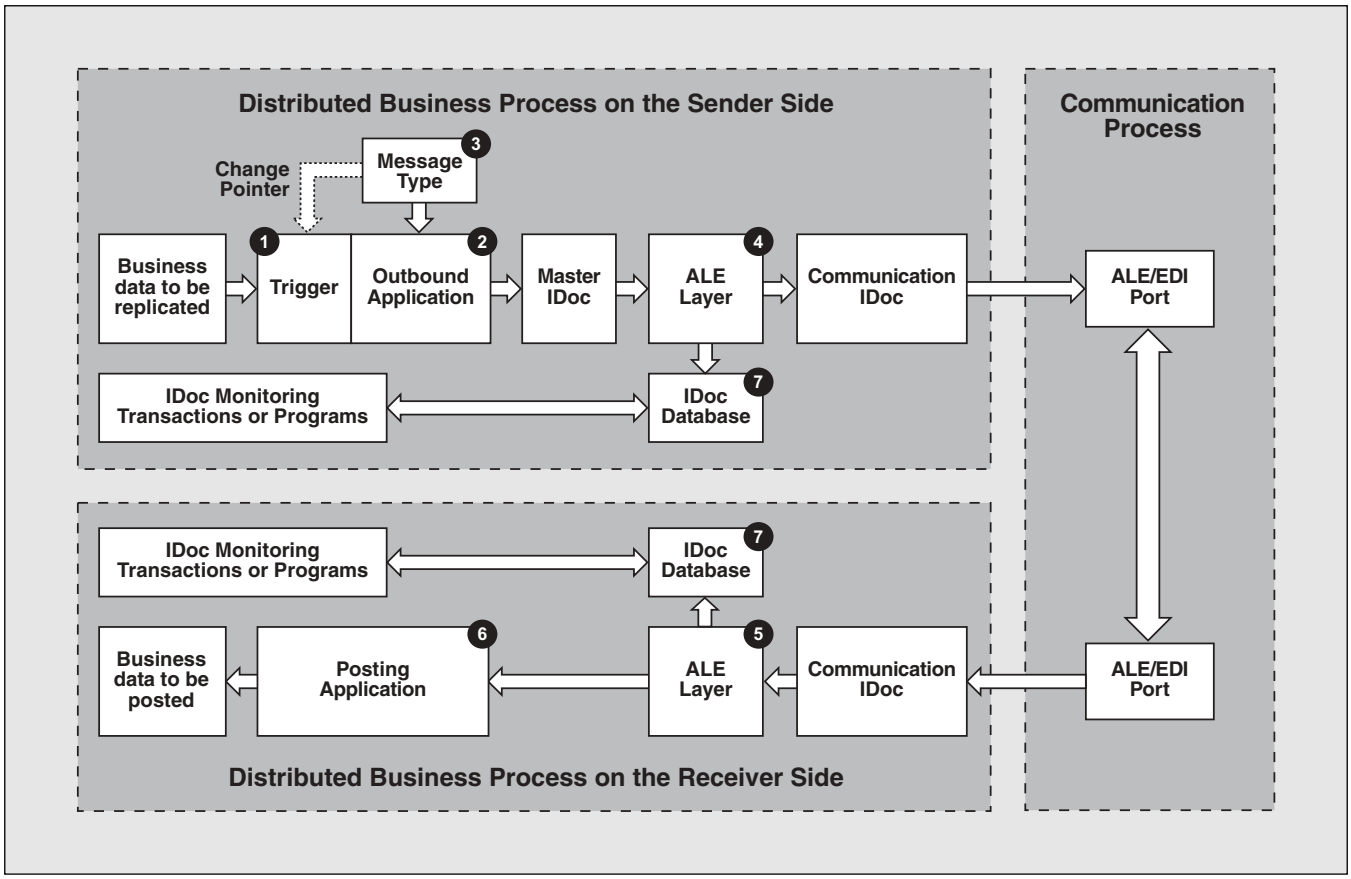
Tools for Minimizing Data Within the ALE Distributed Business Process

SAP provides several tools and features you can use for minimizing the data of an ALE distributed business process. Some features involve reducing the number of IDocs that are created, but the majority involve filtering out the data that is not required. The features you use will depend on the data you have identified as nonessential. In some cases you will have a choice of features and tools, and in other cases you will have only one opportunity for eliminating the data from the process.

Figure 6 depicts the information flow of an ALE distributed business process. It also identifies seven points in this process where you can minimize the data being distributed. In the sections that follow, the discussion of each tool is identified by a number that maps to the relevant point in the diagram.

Minimizing data is most effective in the beginning of the process, because each record and message induces system load at every processing step. The sooner a record or message is eliminated, the less load will be induced at each subsequent step. Putting this

Figure 6 Opportunities for Minimizing the Data of a Distributed Business Process



principle into practice means trying to apply filters as early in the distributed business process as possible — in the trigger (❶) of the outbound application, for example, rather than in the ALE layer (❹). The later you apply a filter, the less effective it is.

There's a small exception to this rule, however. When filtering segments out of a message, look to filters that you can customize (for example, distribution model filters, segment filters, and view filters) over implementing the filter logic in a program exit earlier in the process. A custom filter in the ALE layer (❹), for example, will not consume much more system resources than a program exit filter in the outbound application (❷), since it is just the next processing step, and the maintainability of the interface will also be much better.

In the following sections, you will learn about each of the tools and features you have at your disposal for minimizing data, and where they are best put to use. See my article in the July/August 2003 issue of this publication for a complete explanation of the ALE process and its components, including the filters and user exits discussed here.

Restricting IDoc Creation in the Outbound Program

When looking at the overall business process, the first and most effective place to start minimizing data is at the point where the outbound application is triggered to create an IDoc (❶).

If possible, you should define and implement your selections so that the only IDocs created by the outbound application are those that will be needed for the distributed business process. For an initial

transfer of master data, select only the material master data that is required. For subsequent transfers of changes to the original master database (delta data transfers), restrict the trigger so that only necessary data is transferred.

For example, suppose you have a material master database with a total of 10,000 master data records, and you need to distribute only those master data records (about 500) that are associated with a single specified sales channel. It would be wise to implement a filter in the outbound application's selection program that would apply to the initial data transfer and would pass only the master data belonging to that sales channel. In addition, you would have to implement the same filter conditions in the trigger of the delta data transfer. The final section of this article will show you how to limit the creation of IDocs in the outbound application.

Filtering Records Using Program Exits in the Outbound Application

Whether or not you can use program exits in the outbound application (❷) to filter records depends on the outbound application and the intended purpose of the program exits.

In outbound applications, some program exits — i.e., user exits and Business Add-Ins (BAdIs) — allow you to change the content of all data fields and delete the data records of a message. These exits are the ones you can use for filtering data. Material master BAdI BADI_MATMAS_ALE_CR, for example, allows you to change and delete records of the outbound material master IDocs — you could use it to delete records of the sales view in the material master message based on the content of the procurement type field in the plant view, for instance.

Other exits are designed for implementing a special function and usually can't be used for filtering records. For example, one program exit that you cannot use for directly deleting segments is user exit MGV00002 (Material Master: Read Values for Filter Objects). This user exit reads filter object types and their values from the master records.

To find out if you can use a program exit to filter records, your best course of action is to set up a test implementation of the program exit.

Begin by referring to the documentation (if it exists) of the BAdI or user exit. Then look at the exit's changing or table parameters to see if the data segments that you want to filter have parameters (the names of these parameters often contain strings like EDIDD or IDOC_DATA). If there are no parameters for these data records, then there is no way for you to use the exit to apply filters to them.

If there are parameters for the data records, you then have to find out if the program exit will really support changes to those parameters. Implement the exit with code that deletes one or more of the data records and test it. If after testing you still are not sure whether the exit can delete the segments (for example, you think you have implemented proper program logic but the IDoc still has all its records), debug the program exit and watch the parameter of the data record to see if its data is passed to the application that calls the program exit.

Defining a Subset of the Message Data

With most master data, you can define and implement a data subset of the original message called a "reduced message type" (❸). With this feature, you can ensure that only your defined subset of data will be used to create the master IDoc.

Using a reduced message type can provide an additional performance benefit if you are using the Shared Master Data (SMD) tool to replicate changes in master data. Normally, the fields of an IDoc are related to fields in the business document, and most of these fields — when they are customized as distribution-relevant — act as triggers when changes occur (indicated by the dotted arrow between the message type and the trigger of the outbound application in Figure 6). With a reduced message type, however, only changes to the defined subset of the data — again, only those fields that are customized as distribution-relevant — will serve as triggers for the SMD tool.

Filtering a Message in the ALE Layer of the Outbound Process

The ALE layer of the outbound process (❹) provides three different kinds of filters and a user exit that you can use to restrict the data that is sent.

A **distribution model** filters a segment and its subsegments according to whether the segment's content matches a defined filter value for a specified field (called an "object type"). When applying this type of filter to a field of an IDoc's root segment, you can control the further processing of the whole message. In other words, if a filter condition on a root segment isn't met, then the entire message is neither sent nor stored to the IDoc database. So, if you cannot prevent the creation of an unneeded IDoc in the outbound application, you might at least be able to eliminate it with a filter in the distribution model.

The other two filter types — segment filters and views — can be used to adapt the scope of a message's record structure to the needs of a specific business process.

You can apply a **segment filter** to any nonmandatory segment of a message for a specific sender system and receiver system. Segment filters are static, which means they are not dependent on value-based filter conditions — that is, the segment is always filtered out as long as it belongs to the specified type and partners. Its content is irrelevant. Normally you would use a segment filter on the sender side for filtering segments that are not relevant for a receiver system. For example, a company with an SAP system might send an ORDERS message to a partner with an ERP system that cannot handle text information. In this scenario, the sending company would statically filter all text segments of the ORDERS message with a segment filter. A segment filter can be useful on the receiver side as well, when information (i.e., segments) sent by a partner cannot be handled by the receiving partner.

A **view** is another static filter type. You define a view as a subset of a given message type and assign it to any distributed business process that contains the message type that was parent to the view. The view definition has no relation to the partners of a distributed business process, and views can be used in the outbound process only. Take as an example a company that has to send a shipping notification (DESADV) to various sales companies, some of which have different ERP systems and all of which expect the same message structure. For easier maintenance of the distributed business processes, the company can define a view for the DESADV message and assign it to the partner definitions of the sales companies. All partners will thus get the same subset of the original DESADV message. If changes have to be made later, the company only needs to maintain that one view, which is effective for all of the sales companies. Note that views are processed only by a few outbound processes (e.g., DESADV, INVOIC), however, so their usefulness for minimizing data is very limited.

The remaining option for filtering messages in the outbound process is **user exit ALE00001**. This user exit allows you to delete data records of an IDoc during version control processing. Getting this exit called is a bit tricky. For more information on the conditions under which this user exit is called, please refer to the user exit documentation in transaction *CMOD*, and to my article in the July/August 2003 issue of this publication.

> ✓ ***Note!***
>
> *While Business Add-In IDOC_DATA_MAPPER allows you to change field values in the data segments of all IDocs, you cannot use it to directly delete data segments, because you don't have update access to whole data segments. You can't use it to indirectly delete data segments either, because it is called after the distribution model filter in the outbound process, which means you cannot use it to set values for the distribution model filter. So, the only program exit you can use to delete segments that can be applied to all IDocs is user exit ALE00001.*

Filtering a Message in the ALE Layer of the Inbound Process

At the beginning of the inbound process, the ALE layer (❺) calls user exit ALE00001 for version control. In addition, the inbound ALE layer provides a segment filter that works the same way as the segment filter of the outbound process. The user exit is primarily useful for effecting data transformations, while the segment filter's main function is to take a message the sender regards as "standard" and adapt it to the business process needs of the receiver. However, neither the user exit nor the segment filter is very useful for improving performance, because the reduction of the data volume is limited to the inbound process.

Filtering Records Using Program Exits in the Posting Application

In posting applications (❻), some program exits (user exits and BAdIs) allow you to delete the data records of a message or the structure records of the business object to be posted. For instance, BAdI BADI_MATMAS_ALE_IN allows you to delete records of internal tables that correspond to the structure of material master document data, so you can control data that should not be posted by the material master posting application. Despite their ability to reduce data volume by filtering records, however, these tools have even less of an effect on performance than segment filters in the inbound process.

Archiving and Reorganizing IDocs

All of the IDocs created on a system (whether inbound or outbound) are stored in tables of the database (❼) — e.g., tables EDIDC, EDID4, and EDIDS in SAP R/3 4.x or R/3 Enterprise — so these tables grow constantly as business processes are distributed. Even if the size of these tables is not negatively affecting IDoc processing performance, you can still improve overall performance by reducing their size because IDoc monitoring works faster when these tables are smaller. Besides, there is not much to be gained by collecting data that is no longer needed and is just consuming database space!

For some pointers and ideas on archiving and reorganizing IDocs and other ALE-related data objects, please see the sidebar on the next page and refer to the SAP library.[4]

Deleting IDocs

In some cases, IDocs will no longer be required — in either the database or an archive — after an ALE distribution is completed. These IDocs can be deleted without being archived.

> ✓ ***Note!***
>
> *Before deleting an IDoc, be absolutely certain that you do not need the IDoc data anymore. After the IDoc is deleted, the data is irretrievably gone!*

As of Web AS 6.20, you can use ABAP program RSETESTD (transaction *WE11*) to delete IDocs and, optionally, the data they reference (such as workflow, application log, object relation, or communication data). This program is intended for deleting IDoc data created by SAP test tools, but you can use it to delete any other IDoc data as well. Before using RSETESTD in a production environment, be sure to validate it thoroughly in a test environment.

For Further Information…

This discussion has highlighted the standard ALE features and tools you can use to minimize the data volume of your distributions. Because the tools that allow you to restrict IDoc creation in the outbound application can have the biggest performance impact, the next section provides a more in-depth look at how

[4] Navigate to *mySAP Technology Components → SAP Web Application Server → Middleware (BC-MID) → Application Link Enabling (BC-MID-ALE): ALE Introduction and Administration → Administration of ALE Functions → Optimizing ALE Performance*.

Notes on IDoc Archiving

To be able to use IDoc archiving properly, you need to keep the following points in mind:

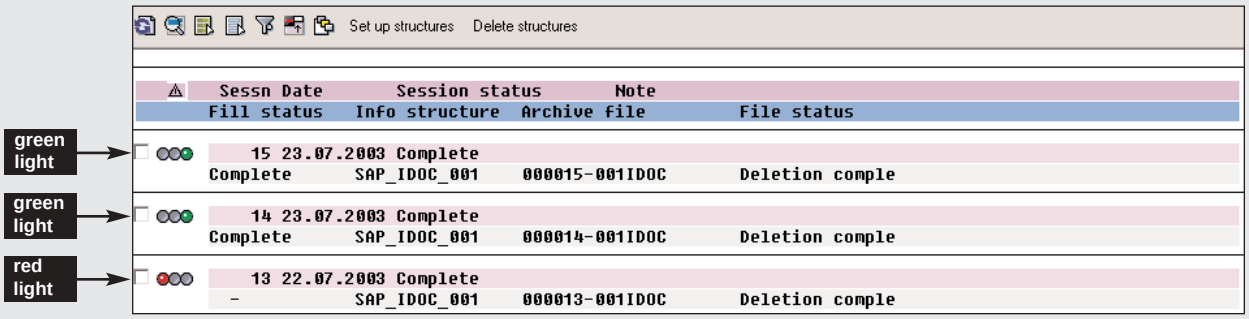
☑ At predefined points in the distribution process, ALE adds status records to an IDoc. It adds them in table EDIDS (IDoc status records) and updates the status field in the IDoc's control record (table EDIDC). In an SAP system, you can decide, for each IDoc status, whether or not an IDoc with this status in its control record can be archived.

☑ To tell the system that a status is relevant for archiving, you must set the archive flag in table STACUST to *X*. Maintain this table with transaction *WE47* (Status Maintenance) and check the field group *Archiving* to see if the status is set to *archiving possible* — some statuses are set to *archiving excluded*, such as status 51 (Erroneous IDocs in SAP standard).

☑ If you want to search for archived IDocs with transaction *WE10* (IDoc Search for Business Content in Archive), you will have to activate the archive info structure* for your IDoc archiving object:

1. To see if the archive info structure for your archiving object IDoc is activated, call transaction *SARJ* (Archive Retrieval Configurator), enter *SAP_IDOC_001* as the standard archive info structure for the archiving object IDoc, choose *Display*, and then press the *Technical data* button. If field *Info structure active* is not flagged in the pop-up screen, press the *Activate* button on the initial screen of transaction *SARJ*. Note that if you have archived any IDocs without activating info structures for them, you will have to set up info structures for those archive files (next step).

2. Still in transaction *SARJ*, navigate to *Environment* → *Create Structure*. A list of the existing archive files will be displayed. Archives without an info structure have a red traffic light icon. Mark those archives in the checkbox to the left of the traffic light icon and press the *Set up structures* button (*F7*) to create an info structure. The screenshot below shows a Create Structure screen with three archive files — two have info structures (the ones with green traffic light icons) and one does not (the one with the red traffic light icon).



3. Once all the info structures are created, you can use transaction *WE10* to search the archived IDocs. Keep in mind that even though the IDocs are archived, each IDoc is represented through one record in the info structure, which is part of the database, and thus the archived IDocs will occupy database space.

* An info structure is a database table that acts as a pointer to an archived data object in the archive file system.

to use them. For more information on using any of the features and tools that can be applied *after* the outbound application is triggered (i.e., program exits, reduced message types, the ALE layer filters, and so forth), please refer to the SAP library and to my previous article in the July/August 2003 issue of this publication.

Restricting IDoc Creation in the Outbound Application

As mentioned earlier in this article, one of the most effective methods at your disposal for reducing the data volume of an ALE distributed business process is the selective creation of IDocs in the outbound application. An outbound application that creates a master IDoc might be triggered by one of several processes. Depending on the process, you have different options for selectively restricting the creation of IDocs.

Figure 7 lists the three master data processes that trigger outbound applications and the means for restricting IDoc creation that correspond with each process. This section provides three detailed discussions of how to minimize data volume when the outbound application has been triggered by one of these processes. Of course, there are other types of outbound applications besides master data processes, but minimizing master data volume is usually the most rewarding way to improve performance — and you may be able to apply the recommendations and tips presented here to other types of outbound applications as well.

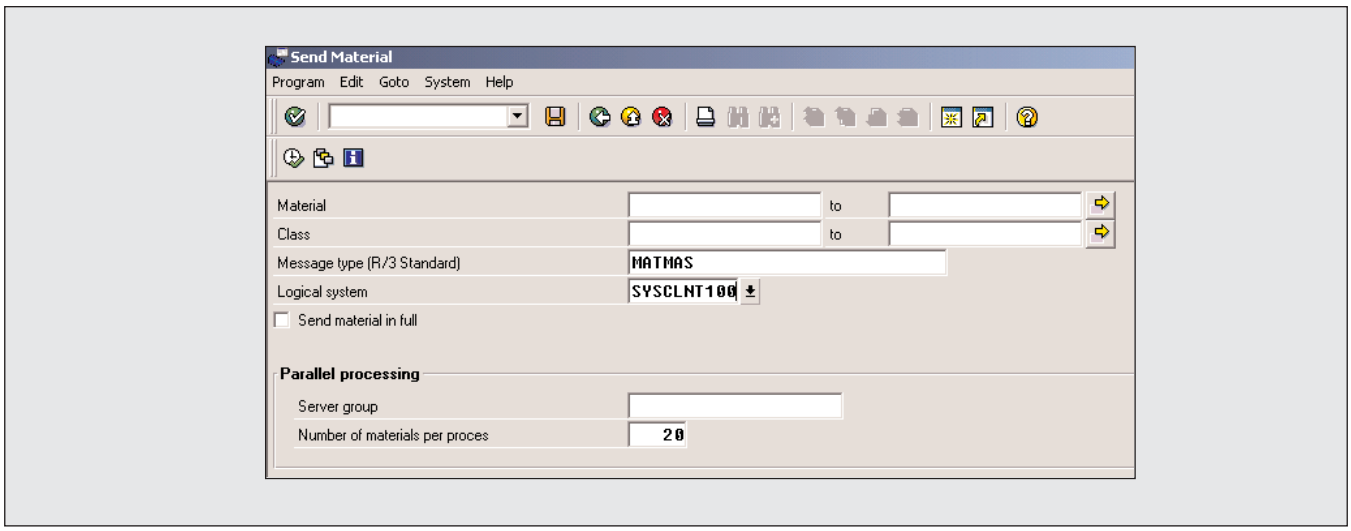
Initial Data Transfer with an Executable Program

When distributing master data, an initial data transfer is usually required to build up the master data basis on the receiver system. In SAP systems, an initial data transfer is usually carried out by executable programs that send the selected master data to the receiver system. Generally, these programs provide select-options for the master data key field, the message type, and the classification. For example, see **Figure 8**, which shows the select-options for Send Material (transaction *BD10*) — material number, class, and message type. Send Customers (transaction *BD12*) provides select-options for customer number, class, and message type.

Figure 7 *Master Data Distribution*

Process That Triggers the Outbound Application		Means to Restrict IDoc Creation
Initial Data Transfer with an Executable Program		• Select-options • Classification • Copying and extending select-options of the standard program
Data Transfer of Changes	Shared Master Data (SMD) Tool	• Reduced message type • BAdI BDCP_BEFORE_WRITE (as of Web AS 6.20) • Copying the standard change pointer processing function module and adding selections
	Workflow Event Linkage	• Field restrictions of events for change documents (transaction *SWEC*)
Request Master Data from the Source System via an Executable Program		• Select-options • Classification • Copying and extending selections of the standard outbound and inbound programs

Figure 8 *Send Material Master IDocs*



There are, of course, many exceptions. A case in point is Send Characteristics (transaction *BD91*), which provides the following select-options: the characteristic's name, validity, message type, and distribution lock. A distribution lock is directly linked to the characteristic's status. For instance, you might customize a characteristic so that when it has the status *In preparation*, it is "locked," and when it has the status *Released*, it is not locked. You could then selectively send characteristics with the status *Released*.

To restrict the creation of IDocs, then, your first step should be to investigate the select-options provided for the master data. If these select-options will not suffice (which is often the case), you will need to judge whether it is worth copying the standard program and adding custom select-options or implementing the distribution model filter for further selections. Copying and extending the standard program does not really enhance the maintainability of the system. I recommend that course only if a huge amount of data would otherwise have to be blocked by the distribution model filter.

Implementing custom select-options is an easy task for an ABAP programmer. Normally a program that sends master data will create an internal table with all selected master data by evaluating the given select-options. This internal table is a kind of "hit list" of all master data to be sent and contains, at a minimum, the master data keys. When adding custom select-options to the program, the programmer simply has to adapt the creation of the hit list or restrict the hit list according to the custom selections.

After an initial data transfer, all changes to the source master data and the newly created master data have to be transferred in order to provide the receiver with the actual data. Depending on the demands of real-time data replication and data-selection requirements, changes might be transferred either by the Shared Master Data (SMD) tool or through SAP Business Workflow event linkage. Both functions provide an opportunity to restrict IDoc creation. The next two examples show you how.

Data Transfer of Changes: The Shared Master Data (SMD) Tool

The Shared Master Data (SMD) tool is based on change pointers, and change pointers are based on change document items.[5] So, change documents are

[5] A change document is identified by an object class (the business object), an object ID (the instance of the business object), and a change ID. The change document consists of a header record (table CDHDR) with general data related to the change, such as date, user, and so forth, and at least one item record (table CDPOS) with information about the changed field.

Figure 9 *Processing Change Pointers*

the basis for SMD data replication. Each change document item that relates to a distribution-relevant field of a message type will result in the creation of a change pointer.

Change pointers are stored either in tables BDCP (change pointer) and BDCPS (change pointer status) or in BDCP2 (a new table for change pointers as of Web AS 6.20 — see the note below).

Transaction *BD21* (program RBDMIDOC, Creating IDoc Type from Change Pointers) will process change pointers for the selected message type in a batch. See **Figure 9** for an example of a *BD21* selection screen that will process change pointers for message type MATMAS.

The delay time for data replication is controlled by the intervals of RBDMIDOC batch job scheduling. When a change pointer is processed by the SMD tool, the change pointer process flag is set in table BDCPS (BDCPS-PROCESS) or BDCP2 (BDCP2-PROCESS).

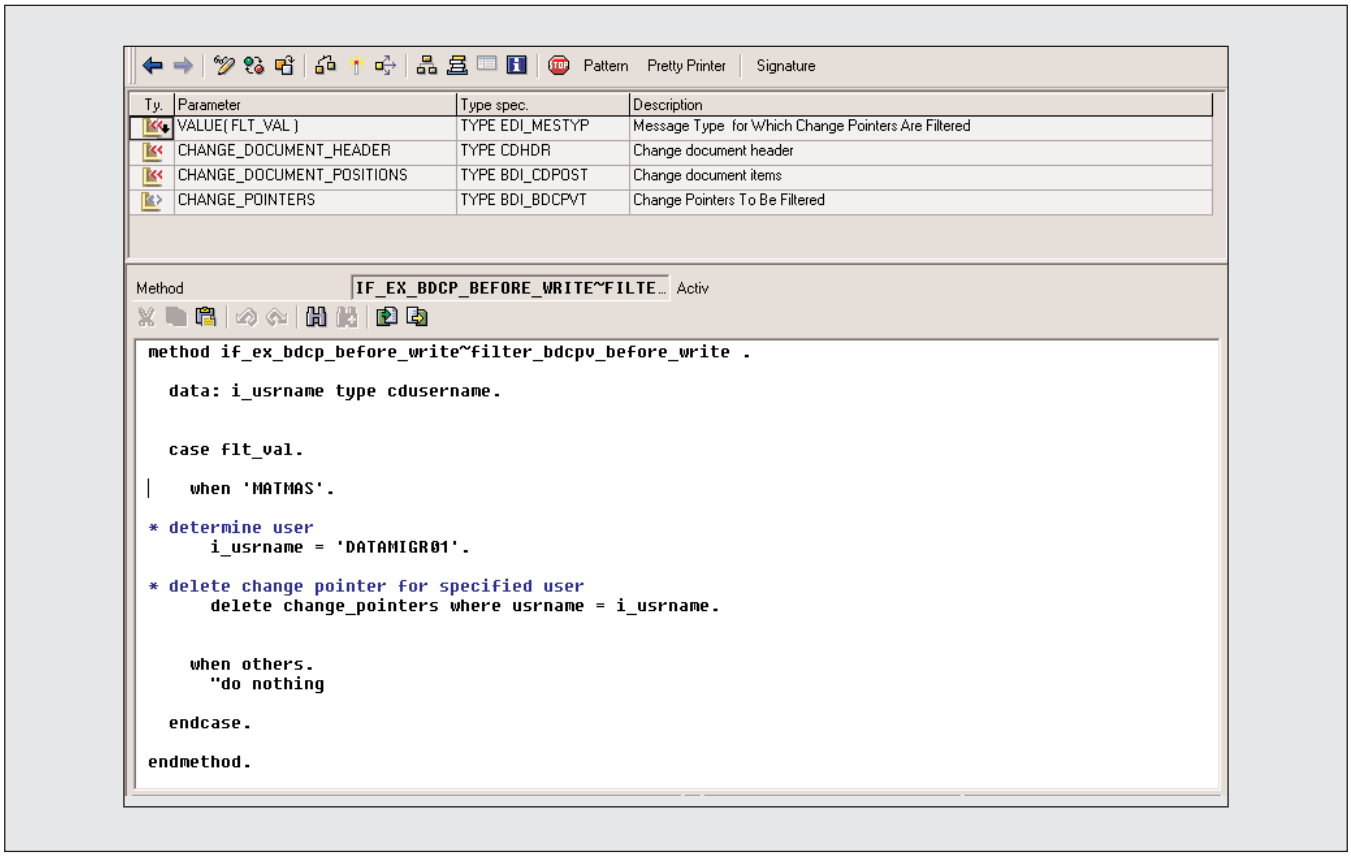
When change pointers are processed, all changes to one instance of a business object are collected into one IDoc per message type, so only the latest change to an instance of a business object within one processing batch will be transmitted. For each distribution-relevant field that has changed, the entire segment that contains the field is transmitted, along with all parent segments and the mandatory segments of the IDoc. Segments that do not contain changed fields and that are neither parent to a segment with a changed field nor mandatory for the IDoc will not be transmitted.

Ultimately, the creation and processing of change pointers does not allow for much selectivity. The only way to customize a subset of trigger fields for change pointers of a whole message is by creating a reduced message type. With a reduced message type, only

✓ ***Note!***

For performance reasons, the change pointer structure has been changed in Web AS 6.20 from a structure with two tables (BDCP and BDCPS, which can be accessed with view BDCPV) to one with a single table (BDCP2). Structures BDCP/BDCPS and BDCP2 coexist. To get an overview of which structure is valid for which message type, look at view V_TBDA2X with transaction SM30 (Extended Table Maintenance). Change pointer processing can be migrated to the new structure with report RBDCPMIG (see SAP Note 305462). Before migration, you must first check to see whether the change pointer processing function module of the message type is able to process the new structure. Field BDCP2SUP in table TBDME (transaction BD60) indicates if a message type's change pointer processing function module is compatible with the BDCP2 structure. With R/3 Enterprise, only a few message types can be migrated to BDCP2. Note that to enhance upgrade performance, you should reorganize all unneeded change pointers before upgrading to Web AS 6.20 (see SAP Note 513454).

Figure 10 A Simple Reference Implementation for BAdI BDCP_BEFORE_WRITE



changes to fields defined in the reduced message type can result in change pointer records. But if a message type is not reducible, if you have to use the standard message, or if you just want to switch off change pointer creation, you will require some other way to gain selective control over change pointer creation or processing.

As of Web AS 6.10, you can implement BAdI BDCP_BEFORE_WRITE to control changes before they get posted to the change pointer tables. See **Figure 10** for a simple reference implementation that suppresses change pointer creation for message type MATMAS for any material master changes that user DATAMIGR01 commits.

✓ ***Note!***

BDCP_BEFORE_WRITE is a filter-dependent BAdI in which the message type serves as the filter value. This BAdI does not allow multiple implementations with the same filter value, however, which means there can be only one active implementation for a filter value. For example, if you have an active BAdI implementation called Z_SELECT_CP_MATMAS with message type MATMAS as the filter value, you cannot activate another BAdI implementation with message type MATMAS as the filter value without first deactivating Z_SELECT_CP_MATMAS. You can, however, implement and activate BAdIs with other message types as filter values.

For SAP Releases 4.6x, you have to do one of the following to gain control over change pointer creation and processing:

- Implement SAP Note 420562, which describes a modification that provides features similar to BAdI BDCP_BEFORE_WRITE (it's a small modification, but it *is* a modification on the standard, nonetheless).
- Create a copy of the message-type-dependent SMD function module (FM) that processes the change pointers, and replace the name of the original FM with the new copy of the FM in field IDOCFBNAME of customizing table TBDME (transaction *BD60*, which controls the SMD processing of FMs per message type). Then you can add coding with your custom selections to the copied FM. You can code your selections right into the FM or define a custom BAdI in the FM and add the selections in implementations of that BAdI. The standard FMs for SMD change pointer processing usually follow the naming convention:

  MASTERIDOC_CREATE_SMD_<mestyp>

  where <*mestyp*> represents the corresponding message type.

> ✓ ***Note!***
>
> *This approach allows you to selectively process change pointers, but it does not affect the creation of change pointers. Therefore, you will have to periodically reorganize the change pointers that have been excluded from processing.*

Data Transfer of Changes: Workflow Event Linkage

Workflow event linkage with master data is based on SAP Business Workflow events that are triggered by change documents. In transaction *SWEC* (Events for Change Documents), you can customize the trigger conditions for events at a fine level of detail by setting field restrictions on the events based on the change document. There, you can specify at the field level whether or not an event should be triggered. You can even define logical conditions — for example, when a new value is different from an old value, or when a value has changed to a specified constant. When your conditions are met in one or more specified fields, the event is triggered.

In event linkage, an event is linked with a receiver function module (a remote-enabled function module) as defined in the event linkage transaction *SWETYPV* (Type Linkages). The event will pass the object ID of the instance of a business object to the function module. The function module calls the ALE outbound application that creates the IDoc. The event linkage interface thus allows you to be highly selective about which IDocs are created, and it works in near-real time. But implementing and maintaining it requires more know-how (including SAP Business Workflow, ALE, and ABAP programming) than implementing and maintaining the SMD tool. Another issue with this interface is that in SAP Business Workflow and ALE, Remote Function Calls (RFCs) are not queued. This means that if you change the same master data twice in quick succession, there is the possibility that the second change will be processed before the first change in the distributed business process, with the result that the second change is overwritten by the first change in the target object. This issue can only be resolved by using the event queue (to make sure that the workflow RFCs are in the right order) and implementing a queued RFC with an inbound queue or by serializing the master IDoc (both features will help get the IDocs processed in the right order).

For more information on the event queue, refer to the online documentation (**http://help.sap.com**) under *SAP NetWeaver Components → SAP Web Application Server → Business Management* (*BC-BMT*) → *WebFlow Engine (BC-BMT-WFM) → Reference Documentation → Workflow System Administration → Event Manager Administration → Event Queue Administration*. For information on queued RFCs, go to *SAP NetWeaver Components → SAP Web Application Server → Middleware (BC-MID) →*

Remote Function Call (BC-MID-RFC) → Queued Remote Function Call (qRFC). For details on IDoc serialization, see *SAP NetWeaver Components → SAP Web Application Server → Middleware (BC-MID) → Application Link Enabling (BC-MID-ALE) → ALE Introduction and Administration → Administration of ALE Functions → Serialization of Messages*.

For a good introduction to setting up interfaces based on SMD or SAP Business Workflow event linkage, see Amy Stapleton's article "Real-Time, Outbound Interfaces to Non-R/3 Systems Made Simple with Change Pointers, Message Control, and Workflow" in the Premiere Issue of this publication.

Selective Retrieval of Master IDoc Data — Request Master Data from the Source System via an Executable Program

The previous two scenarios — initial data transfer and transfer of changes — represent a "push" technology, because the source system that contains the data to be distributed "pushes" the data to the receiver system. Now we'll look at SAP's "pull" technology for master data, where the receiver system can request data from the source system.

In this scenario, a request IDoc is sent to the source system. The request IDoc transmits the requested message type and the select-options of the master data to be requested. The source system processes the request IDoc and returns the requested IDocs. The programs for requesting master data usually provide select-options for the master data key field and for classification — for example, transaction *BD11* (Get Material) provides select-options for material number and classification, and transaction *BD13* (Get Customers Master) provides select-options for customer number and classification.

When restricting the data that is sent in this scenario, it's best to use the program's standard select-options. Enhancing the program's interface by adding custom selections can be quite awkward. While the request IDoc itself can transmit any select-options, you have to adapt the program on the sender side to add new select-options. Likewise, you have to adapt the program on the source system side to enable it to process the additional selection criteria. In some cases, however, you'll want to enhance the program's interface, so here I will show you how.

> ✓ ***Note!***
>
> *Adding custom select-options to a program's interface requires some familiarity with ABAP development. If you do not have sound ABAP skills, ask an ABAP programmer to assist you with the steps in this section.*

Follow these steps to enhance a program's interface with additional select-options:

1. Using transaction *SE38* (ABAP Editor) or *SE80* (Object Navigator), create a new program by copying the original sending program to the customer namespace. Add the desired select-options to the new program and add each select-option's record to the internal table that gathers the selection data for the request IDoc. For example, T_REQIDOC is the internal table for the program that requests material master data (RBDFEMAT). **Figure 11** shows a record from T_REQIDOC for a material master select-option with an object value of MATNR. Each record in T_REQIDOC is represented in the request IDoc as one segment of type E1ALER1.

2. Create a new function module on the system that will receive the request IDoc by copying the inbound function module that processes the request IDoc on the source system side — e.g.,

Figure 11 A Select-Option Record

No.		Component name	Type	Lngth	Contents
1		OBJVALUE	C	70	MATNR
2		SIGN	C	1	I
3		OPTION	C	2	BT
4		LOW	C	40	000000000000000042
5		HIGH	C	40	000000000000000046

Figure 12 Define the Characteristics of the Newly Created Inbound Function Module

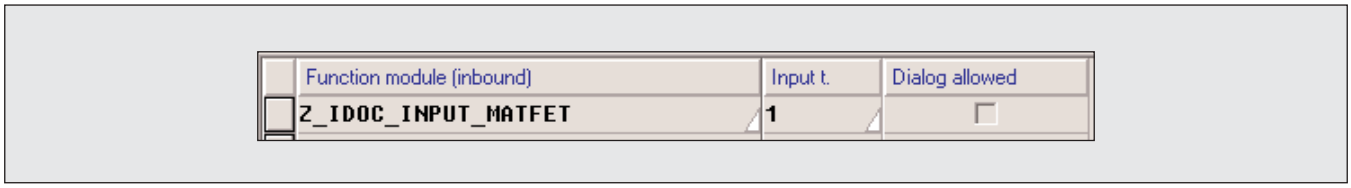


Figure 13 Assign a Message Type

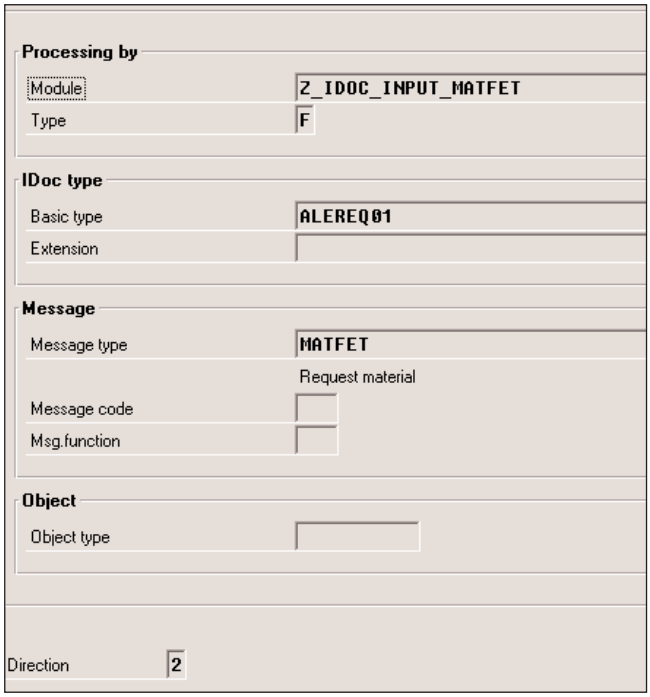
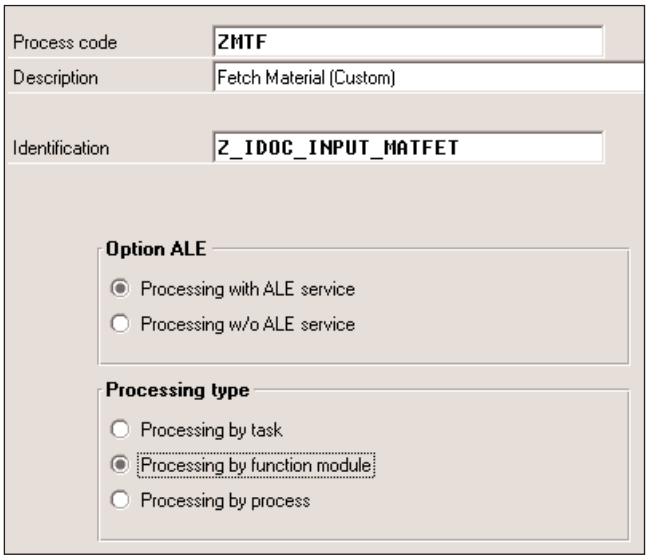


Figure 14 Create a New Process Code



copy IDOC_INPUT_MATFET, which requests the material master, to customer namespace Z_IDOC_INPUT_MATFET. Adapt the coding to the additional selection criteria. Don't forget to copy the top include entries and the relevant user-defined include files of the function group.

Next we need to customize the control tables that enable the newly created inbound function module to be used in the ALE inbound process — i.e., we must define a new process code for the extended request inbound function module. Follow these steps:

1. Define the processing characteristics for the newly created inbound function module. In transaction *SM30* (Extended Table Maintenance) create an entry for the new function module in view V_TBD51 (Characteristics of Inbound Function Modules), as shown for inbound function module Z_IDOC_INPUT_MATFET in **Figure 12**. Give the new function module the same "input type" and "dialog allowed" attributes as the original.

2. Assign the newly created function module to a message type. In transaction *WE57* (Assign Function Module to Logical Message and IDoc Type), copy the settings of the original function module to the new function module (*Z_IDOC_INPUT_MATFET* in **Figure 13**).

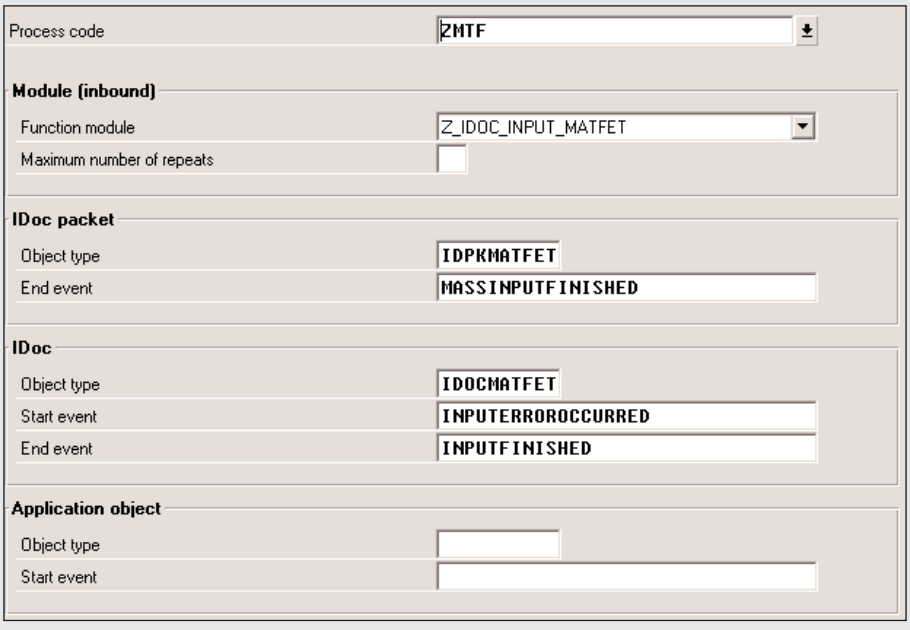
3. Define a new process code for the extended request inbound process. In transaction *WE42* (Inbound Process Code), create a custom process code and give it a name and a description — *ZMTF* and *Fetch Material (Custom)* in **Figure 14** — and then duplicate the settings of the original inbound process code.

4. Associate the new process code (*ZMTF* in the example) with the inbound function module and object definitions (see **Figure 15**). Using

Helpful Hints

- ☑ For a sound overview of an IDoc's segments and fields, browse through the IDoc structure using transaction *WE30* or *WE60*.
- ☑ Use the syntax check of the test tool for processing IDocs (transaction *WE19*) to verify that the minimized IDoc retained the appropriate structure.
- ☑ Create and change instances of the target business object with transaction *WE19* to determine whether the minimized IDoc will support the desired distributed business process.
- ☑ If you require read access to archived IDocs, be sure to create archive info structures for those archive files, which will allow you to use transaction *WE10* to search the archived IDocs for control and business data. Keep in mind that creating archive info structures consumes database space.
- ☑ For more information about ALE:
  - Refer to previously published *SAP Professional Journal* articles on ALE (see **www.SAPpro.com**).
  - The SAP Service Marketplace (**http://service.sap.com/netweaver**) contains excellent presentations in the ALE and EDI section. To download the presentations, choose *Application Link Enabling* from the *Related Technology Topics…* dropdown list.
  - Search the SAP Notes for ALE-related topics.
  - Refer to the SAP online help at **http://help.sap.com** under *SAP NetWeaver Components → SAP Web Application Server → Middleware (BCMID) → Application Link Enabling (BC-MID-ALE)*.

Figure 15 Associate the Process Code with the Inbound Function Module and Object Definitions



transaction *SM30* (Extended Table Maintenance), copy the control information of the original process code to table TBD52 (Function Modules for Inbound ALE-EDI).

Conclusion

If you take one lesson away from this article, let it be this: You will get the biggest performance gains by minimizing data volume at the beginning of the distributed business process. Whenever possible, create data transfer messages (IDocs) only for the data that must be transferred. Then filter out all segments of the created messages that you do not need.

This two-part article series has provided you with a good foundation for optimizing the processing of an ALE data distribution and reducing the system load by minimizing the data to be processed. To find further opportunities for improving performance, I recommend searching the SAP Notes database. It contains many practical tips on ALE performance.

Optimizing ALE distributed business processes is a complex task. To ensure your success in setting up and maintaining a stable and efficient ALE distribution, my highest recommendation is to assemble a distribution team that possesses all the requisite technical skills as well as good communication skills.

Arthur Wirthensohn is a senior consultant at EDS Switzerland and a member of EDS's international Technical Leadership Network. He has worked both as a project manager and product manager in the ERP and IT integration business for many years, mainly in the retail, consumer products, manufacturing, and trade industries. For the past two years, Arthur was the product manager responsible for Application Services for SAP systems. Currently, he is the project manager and technical lead of the Enterprise Application Integration (EAI) department, which delivers SAP-related services such as ALE/EDI, SAP Business Workflow, Web Application Server, Data Migration, and ABAP Programming Services. Arthur can be reached at arthur.wirthensohn@eds.com.

