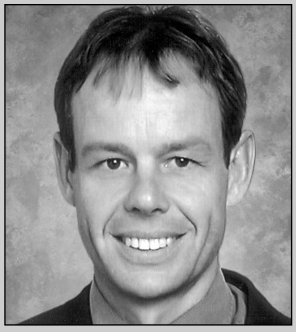
Understanding and Optimizing Your ALE Data Distribution: Controlling ALE Processing

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.

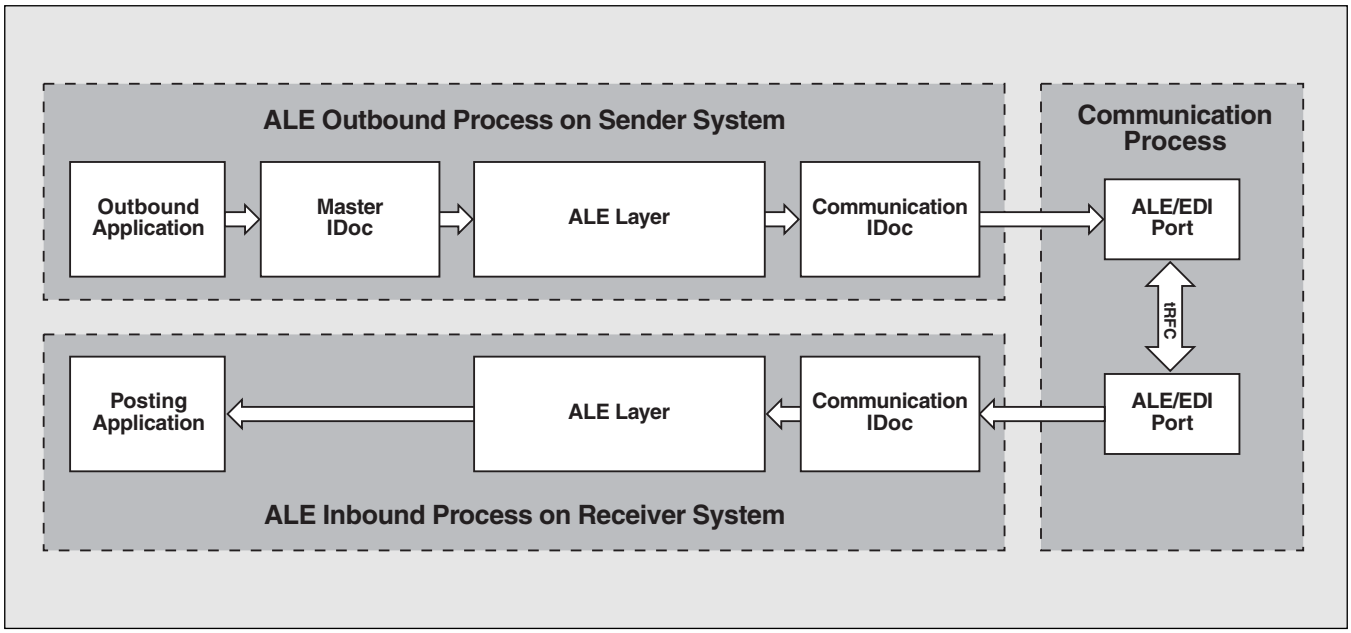
(complete bio appears on page 26)

As the integration layer inherent in every SAP Basis system, Application Link Enabling (ALE) provides proven features for distributing your data across integrated systems. It is the standard SAP tool of choice for linking an SAP system to another SAP system, to an application integration system like the SAP Exchange Infrastructure (XI), or to an EDI (electronic data interchange) subsystem.[1] No matter how good a data distribution tool is, however, there is one inescapable fact: the more data you distribute, the more you need to focus on performance and capacity issues. If these issues are not properly addressed, distributing data can cause reduced system availability and poor performance for dialog users. In a worst-case scenario, an ALE data distribution can monopolize the resources of an entire SAP system, preventing dialog users from executing business transactions or even logging on to the system. Such effects will not endear your data distribution activities to other users of the SAP system!

How can you — as a programmer, project manager, system administrator, or technical consultant — optimize ALE data distributions so that they perform well without monopolizing system resources? You will need to understand 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. With this knowledge, you can maximize the efficiency, timeliness, and stability of your ALE processes without unduly straining system resources.

[1] Behind this integration layer, the outbound process uses BAPIs or ordinary ABAP code to extract data and provide it to the ALE layer, while the inbound process uses function modules, BAPIs, or SAP Business Workflow tasks to receive and handle the data (e.g., post a material master). Within ALE, BAPIs can be used as encapsulated program code for extracting (i.e., replicating) business data on the sender side and posting it on the receiver side.

Figure 1 Information Flow in an ALE Distributed Business Process



This article provides you with a foundation for that understanding, taking you all the way through the ALE data distribution process and showing you where opportunities exist for controlling and optimizing it. By the time you have finished reading, you will have a list of specific recommendations to follow and, more important, you will be able to devise your own optimization techniques and strategies in a variety of circumstances.

An Overview of the ALE Distributed Business Process

In order to optimize your ALE data distributions, it helps to understand what happens at each stage of the process, from extracting business data out of the sender system to transferring that data to the receiver system and posting it. We'll begin with an overview of the ALE distributed business process and then dive into each stage to take a detailed look at opportunities for optimization.

The diagram in **Figure 1** shows, in a simplified form, the way information flows in an ALE distributed business process:

- ALE data distribution starts with the outbound process on the sender system, where business data is replicated from the source database and transformed into a message container called an "IDoc" (Intermediate Document). This initial IDoc is called the "master IDoc."
- After the outbound application has created a master IDoc, the ALE layer determines certain control information, such as the receiver system and the communication technology, performs version management, and applies data transformations like filters or conversion rules.
- The ALE layer then creates a "communication IDoc" that contains both the information from the master IDoc and all the transformations that were applied during the outbound process.
- Next, the IDoc is transmitted to the receiver system through the communication process, which performs the physical transportation of the IDoc. ALE provides various options for performing the communication between

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

the sender and receiver systems, each controlled by a different type of port,[2] which specifies the technical properties of the communication. Here, we will focus on the tRFC[3] (transactional Remote Function Call) port type. tRFC is the data transfer technology best suited for communication between SAP systems, or, for example, between an SAP system and an external application integration system, so whenever possible, use tRFC to connect systems.

> ✓ ***Note!***
>
> *The recommendations in this article are also valid for ALE processes that use port types other than tRFC for data transfer, such as writing or reading IDoc data to or from a file system (using the "File" port type); writing an XML file to a file system (using the "XML-File" port type); transferring data via HTTP (using the "XML-HTTP" port type); or processing IDoc message data with ABAP custom coding (using the "ABAP" port type).*

- The inbound process receives the communication IDoc from the communication process.
- After receiving the IDoc, the ALE layer performs any necessary transformations (applies data filters, conversions, etc.).
- Finally, the ALE layer passes the IDoc to the posting application, which posts the IDoc data to the destination database on the receiver system as an instance of a business object (a material master, for example).

The Importance of Dialog Work Processes

The availability of dialog work processes is a very important factor in SAP system performance — both users and system processes that call remote function modules (like ALE, SAP Business Workflow, parallel processing, and so forth) consume dialog work processes, and sometimes compete with one another for those processes. (For a discussion that puts dialog work processes into the context of work processes in general, see the sidebar "A Short Explanation of Work Processes in the SAP System" on the next page.)

As we walk through the ALE distribution process in the sections that follow, we'll pay special attention to the consumption of dialog work processes and how that consumption can be controlled for the benefit of both users and other processes that require them.

Optimizing Performance — The ALE Outbound Process

The ALE outbound process consists of two main steps (refer back to Figure 1):

1. The outbound application creates a master IDoc that contains the business data to be transmitted.
2. The ALE layer determines the recipient system, performs version control, and applies any filters and conversions.

---

[2] In SAP terminology, the term *port* refers to the technical communication process used in an ALE data distribution scenario or EDI process. As of SAP Web Application Server 6.20, ALE supports the following port types: *tRFC* (transactional Remote Function Call); *File* (IDoc data transmitted as a file to or from a file system); *CPIC* (Common user Programming Interface Communication); *ABAP* (IDoc data transmitted to or from a function module); *XML-File* (XML data transmitted as a file to a file system); and *XML-HTTP* (XML data transmitted via HTTP).

[3] SAP's RFC (Remote Function Call) technology is a proprietary interfacing protocol for calling program logic from a remote system. There are different kinds of RFCs — e.g., synchronous (sRFCs), asynchronous (aRFCs), transactional (tRFCs), and queued (qRFCs). For more information, refer to the SAP library under *SAP NetWeaver Components → SAP Web Application Server → Middleware → Remote Function Call*.

A Short Explanation of Work Processes in the SAP System

In an SAP system, services are executed through their corresponding work processes. For example:

- A dialog service, which processes user input, workflow, or Internet communication, for instance, is executed by a DIA work process. A user logged on to an SAP system occupies at least one dialog work process when interacting with the system.
- A batch (or "background") service, which processes programs that are launched by the job scheduler, is executed by a BTC work process.
- An update service, which makes asynchronous changes to the database, is executed by a UPD work process.
- An enqueue service, which handles the locking mechanism, is executed by an ENQ work process.
- A spool service, which manages output such as printing, is executed by an SPO work process.

Defining Work Processes and Their Distribution

System administrators initially define the types and numbers of work processes to be used in the **instance profile** of each server installation (instance) of the system.

The SAP system's **operation mode** feature contains settings that define the distribution of work processes during a specified time interval. Using the operation mode settings, you can adjust the number of work processes in the system according to specific requirements — for example, the different processing needs of daytime (comparatively more dialog) and evening (comparatively more batch) processing. Note that operation mode settings can only control the *distribution* of the work processes

Step 1: The Outbound Application Creates a Master IDoc

The master IDoc is created by the outbound application, which can be any piece of ABAP code that extracts business, customizing, or control data and transforms that data into an IDoc structure. **Figure 2** outlines the most common types of outbound applications and indicates whether or not they support parallel processing. As we will see later in the article, parallel processing offers many options for improving system performance, so it is helpful to know which outbound applications support this functionality.

The outbound application creates a master IDoc and passes it to the ALE layer, where the IDoc data will be written to the IDoc database tables after the distribution model filter is applied. The ALE layer adds a status record to table EDIDS (IDoc Status Records) after each step of the process. Once the IDoc is stored in the IDoc database tables, the ALE layer adds status record 01 (IDoc created) to the IDoc's EDIDS table.

✓ ***Note!***

To help you learn to follow IDoc processing in your own ALE distributed business processes, this article discusses the status records most relevant to an ALE scenario using an RFC connection.

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

defined in an instance profile; unlike the instance profile settings, they cannot control the total number of work processes being used. Operation mode settings are not mandatory for getting an SAP system up and running, but SAP recommends using them, and for some system activities, such as a release upgrade, they are required.

Maintaining the Usage of Dialog Work Processes

The function module called by a Remote Function Call (RFC) is *always* processed by a dialog work process, regardless of whether the program that calls it was started in dialog or batch mode. Therefore, because RFCs are used in so many types of applications — parallel processing, ALE communication, SAP Business Workflow, and so on — there is always the danger of a dialog work process shortage for other dialog users.

Using the instance profile settings, you can restrict the number of dialog work processes occupied by RFCs. You can set dialog work process **quotas** for RFCs or define an absolute number of dialog work processes that must not be occupied by RFCs (note that these restrictions apply only to asynchronous and transactional RFCs). Setting quotas for RFCs requires a sound understanding of the SAP Basis environment, and, since not all RFC types follow these settings, finding the correct parameter values is usually a matter of trial and error. I recommend that you talk with your system administrator about your ALE dialog work process needs and refer to SAP Note 142419 (RFC load distribution via Quota settings) before taking this approach.

If you are using parallel processing, there is an additional option for maintaining the usage of dialog work processes: you can create a **server group**, which allows you to define dialog work process resources on specific servers for parallel processing. Note that you can use a server group only in applications that are programmed to support parallel processing.

Figure 2 The Most Common Types of Outbound Applications

Outbound Application Type	Example	Parallel Processing Supported?
An online program with select-options for the business data to be sent	• Transactions for sending master data, such as BD12 (Send Customers) • Specific ABAP programs for sending business documents, such as RFFOEDI1 (Payment Orders by EDI)	This depends on the program; most programs for sending master data provide a parallel processing option
An application that issues output in the SAP R/3 modules MM (Materials Management) and SD (Sales and Distribution)	• Processing table NAST with ABAP program RSNAST00 to issue output via ALE or EDI	No

(continued on next page)

Figure 2 (continued)

Outbound Application Type	Example	Parallel Processing Supported?
An application that links an SAP Business Workflow event to a remote function module that creates an IDoc	• When a customer master is changed, the change event passes the customer master number to a remote function module that creates a customer master IDoc	No
An application that processes change pointers*	• Processing change pointer table BDCP through the Shared Master Data (SMD) tool (transaction BD21)	No
A program-exit call	• User exit MB_CF001 or Business Add-In MB_DOCUMENT_BADI creates a stock movement IDoc for an external warehouse system	No
A synchronous or asynchronous call from an external system	• An SAP BW (Business Information Warehouse) extractor sends a request IDoc to an R/3 system, which answers with a data IDoc	No

* For further details on using change pointers, see the 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.

> ✓ ***Tip***
>
> *The IDoc monitoring transactions — e.g., WE02 (Display an IDoc) and BD87 (Reprocess IDocs in error or waiting for action) — allow you to navigate through all of an IDoc's status records.*

Step 2: The ALE Layer Determines the Recipient System, Performs Version Control, and Applies Any Filters and Conversions

In the second step of the outbound process, the ALE layer determines the recipient (or recipients) of the data being exported, performs version management (which guarantees the compatibility of IDoc messages

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

in different SAP releases), and applies any filters and conversions. At the end of a successful outbound process, status record 30 (IDoc ready for dispatch) is added to the IDoc's status record table; otherwise, status record 26 (Error during syntax check of IDoc) or status record 29 (Error in ALE service) is added. Depending on the *output mode* parameter,[4] the IDoc will either be transferred immediately through the communication process or remain in the database and be collected and transferred at a later time, by a processing program like RSEOUT00 (Process all selected IDocs) or via transaction *BD87* (Reprocess IDocs in error or waiting for action).

The Use of Work Processes in the Outbound Process

In the outbound ALE process, the use of work processes is highly dependent on the outbound application's support for parallel processing. Programs without a parallel processing option create IDocs sequentially, one after the other, using at least one work process, though RFCs called out of the sending program's main processing might use additional dialog work processes. (Remember from the sidebar on pages 6-7 that the function module called by an RFC is *always* called by a dialog work process, regardless of whether the program that calls it was started in some other mode.)

Take a look at the third column in Figure 2. The programs capable of parallel processing are the ones with select-options for the business data to be sent — the ones that send masses of master data over an ALE interface (e.g., for an initial data transfer). Because of the potentially huge data volumes involved, these programs are especially predisposed to benefit from the performance and processing control offered by parallel processing.

You can optimize the resource use of programs that support parallel processing using *server groups*. A server group consists of one or more instances (practically speaking, servers that provide dialog work processes). Using a server group, you can specify which servers, along with their dialog work process resources, are to be used for parallel processing — for example, you could include all application servers in a server group but exclude the database server to protect it from resource shortages. You can also specify within a server group certain resources for each instance — for example, you could define the minimum number of dialog work processes that must not be occupied by the server group or the maximum percentage of dialog work processes that can be occupied by the server group. To create or maintain a server group, select *CCMS → Administration → System Administration → Network → RFC destinations* or call transaction *RZ12*.

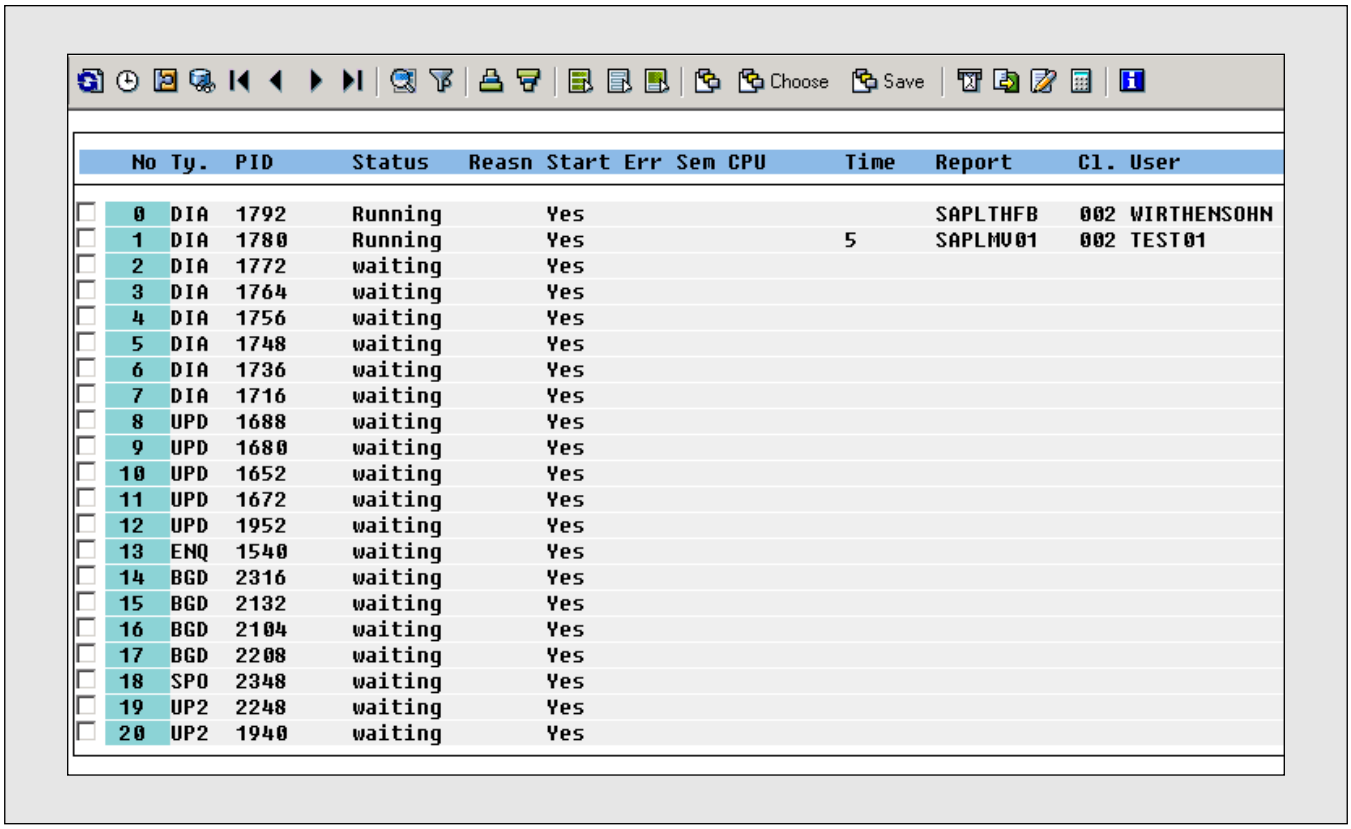
Generally, I recommend using the parallel processing option with a server group even if the server group will contain just one instance — you can still control the amount of dialog work processes your single instance consumes.

Parallel processing is the only option for optimizing work processes in the outbound process. Programs that do not support parallel processing must be started when enough time and system resources are available (or they must be scheduled for such a time), so that other user interactions or parallel processing programs are not impeded. In the next section, we will look at how the outbound process allocates work processes when started in different modes.

For more information on parallel processing and server groups, please refer to Susanne Janssen and Werner Schwarz's article "Speed Up High-Throughput Business Transactions with Parallel Processing — No Programming Required!" in the January/February 2002 issue of this publication.

[4] The *output mode* parameter defines how IDocs will be transmitted to the receiver system in terms of "coupling" (which will transfer an IDoc immediately) or "decoupling" (which will collect IDocs and transfer them later) the outbound process from the communication process. If the File or XML-File port type is used, the output mode also controls the start of any subsystems.

Figure 3 Creating IDocs in Dialog Mode Without Parallel Processing



No	Ty.	PID	Status	Reasn	Start	Err	Sem	CPU	Time	Report	Cl.	User
0	DIA	1792	Running		Yes					SAPLTHFB	002	WIRTHENSOHN
1	DIA	1780	Running		Yes				5	SAPLMV01	002	TEST01
2	DIA	1772	waiting		Yes							
3	DIA	1764	waiting		Yes							
4	DIA	1756	waiting		Yes							
5	DIA	1748	waiting		Yes							
6	DIA	1736	waiting		Yes							
7	DIA	1716	waiting		Yes							
8	UPD	1688	waiting		Yes							
9	UPD	1680	waiting		Yes							
10	UPD	1652	waiting		Yes							
11	UPD	1672	waiting		Yes							
12	UPD	1952	waiting		Yes							
13	ENQ	1540	waiting		Yes							
14	BGD	2316	waiting		Yes							
15	BGD	2132	waiting		Yes							
16	BGD	2104	waiting		Yes							
17	BGD	2208	waiting		Yes							
18	SPO	2348	waiting		Yes							
19	UP2	2248	waiting		Yes							
20	UP2	1940	waiting		Yes							

> ✓ ***Tip***
>
> *When specifying dialog work process resources in a server group, try to occupy as many processes for the ALE outbound process as needed, but leave enough vacant for other dialog work process consumers, such as dialog users, SAP Business Workflow, general ALE functions, the Internet Communication Manager (e.g., for HTTP processing), and the like.*

Determining the Work Process Resources Used by the Outbound Process

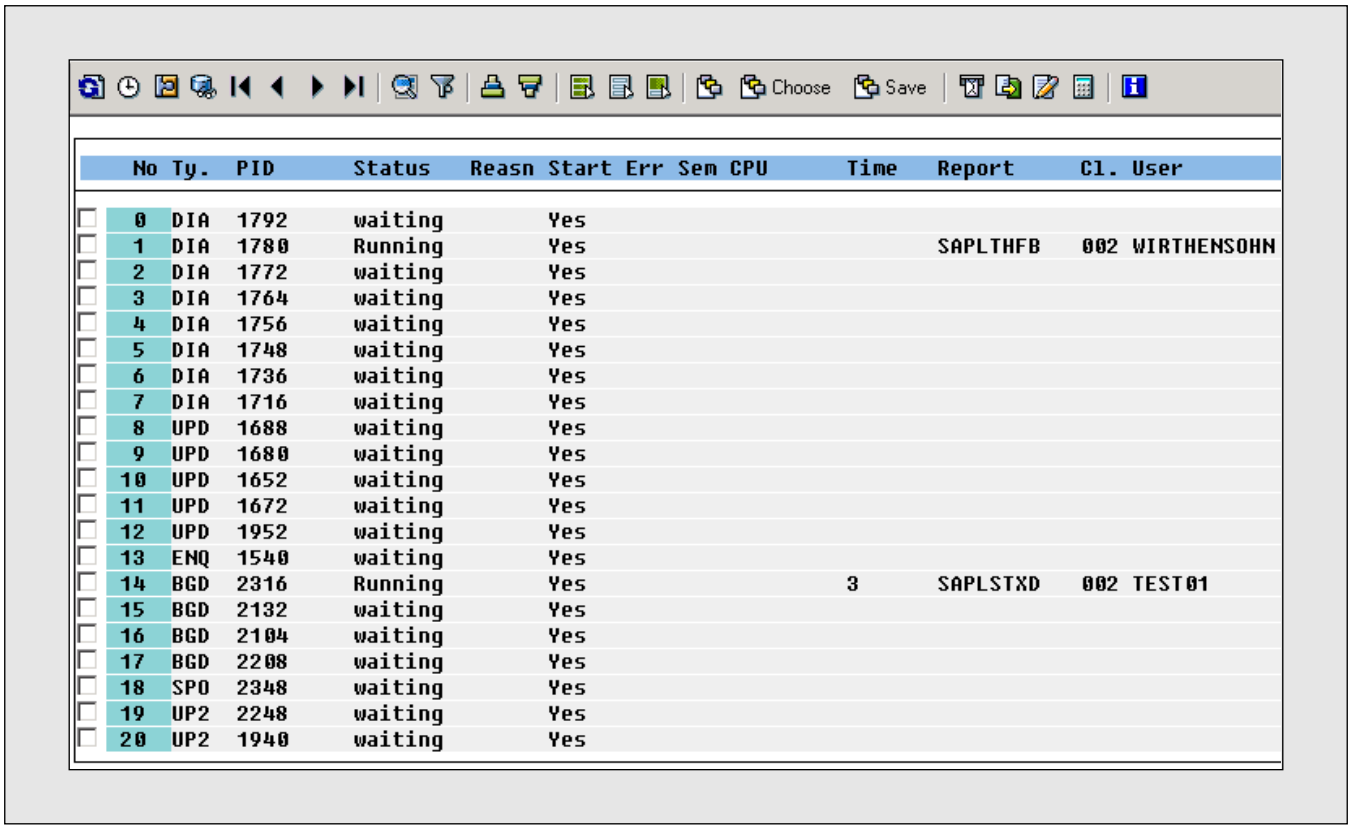
To optimize the ALE outbound process, you need to understand how the sending system processes the tasks of an outbound application in dialog mode or background (batch) mode, both with and without parallel processing. To identify the resources that are used when the sending system creates an IDoc during the outbound process, we'll look at four different scenarios in which we will create a material master IDoc using program RBDSEMAT (Send Material) and the following modes:

- Dialog mode without parallel processing
- Background (batch) mode without parallel processing
- Dialog mode with parallel processing
- Background (batch) mode with parallel processing

For each scenario, we will examine and compare the Process Overview screen (transaction *SM50*),

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

Figure 4 Creating IDocs in Background (Batch) Mode Without Parallel Processing



No	Ty.	PID	Status	Reasn	Start	Err	Sem	CPU	Time	Report	Cl.	User
0	DIA	1792	waiting		Yes							
1	DIA	1780	Running		Yes					SAPLTHFB	002	WIRTHENSOHN
2	DIA	1772	waiting		Yes							
3	DIA	1764	waiting		Yes							
4	DIA	1756	waiting		Yes							
5	DIA	1748	waiting		Yes							
6	DIA	1736	waiting		Yes							
7	DIA	1716	waiting		Yes							
8	UPD	1688	waiting		Yes							
9	UPD	1680	waiting		Yes							
10	UPD	1652	waiting		Yes							
11	UPD	1672	waiting		Yes							
12	UPD	1952	waiting		Yes							
13	ENQ	1540	waiting		Yes							
14	BGD	2316	Running		Yes				3	SAPLSTXD	002	TEST01
15	BGD	2132	waiting		Yes							
16	BGD	2104	waiting		Yes							
17	BGD	2208	waiting		Yes							
18	SPO	2348	waiting		Yes							
19	UP2	2248	waiting		Yes							
20	UP2	1940	waiting		Yes							

which displays the typical usage of work processes by an outbound process.

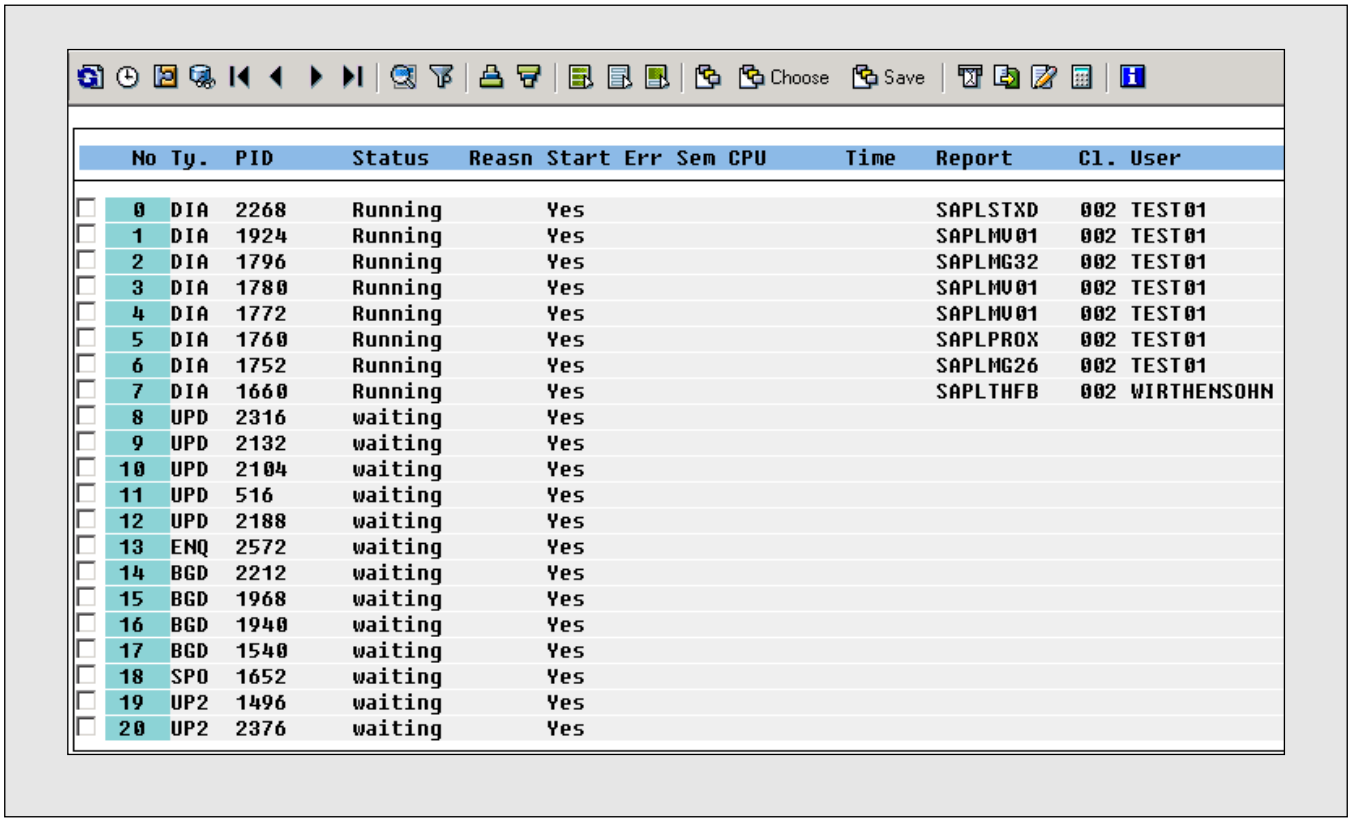
> ✓ ***Note!***
>
> *To better distinguish between the processes being used for the monitoring transaction (SM50) and those being used for the outbound process, in each scenario user WIRTHENSOHN is logged on to the system as a monitoring user occupying one dialog work process (transaction SM50), and user TEST01 is calling the master data-sending program (the outbound process).*

Here's what happens when program RBDSEMAT is called in each scenario:

- **Dialog mode without parallel processing:** Here, the IDocs are created sequentially. In the Process Overview for this scenario (**Figure 3**), you can see that one dialog process is being used by *TEST01* — the process that creates the IDoc. (Note that the "type" column shows the type of work process used.) Of course, there could be additional dialog work processes occupied by RFCs in the main sending program (these are not shown in Figure 3).

- **Background (batch) mode without parallel processing:** The IDocs are created sequentially in this scenario as well (**Figure 4**); as before, *TEST01* is using one process (a batch process this time). Here again, additional dialog work processes could be involved — in this case, in the processing of an RFC in the batch process (not shown in Figure 4).

Figure 5 Creating an IDoc in Dialog Mode with Parallel Processing

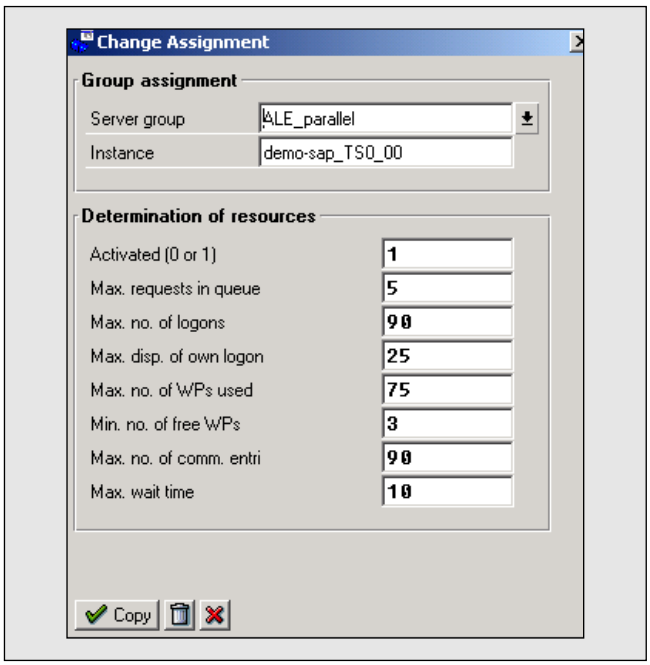


No	Ty.	PID	Status	Reasn	Start Err	Sem CPU	Time	Report	Cl.	User
0	DIA	2268	Running		Yes			SAPLSTXD	002	TEST01
1	DIA	1924	Running		Yes			SAPLMV01	002	TEST01
2	DIA	1796	Running		Yes			SAPLMG32	002	TEST01
3	DIA	1780	Running		Yes			SAPLMV01	002	TEST01
4	DIA	1772	Running		Yes			SAPLMV01	002	TEST01
5	DIA	1760	Running		Yes			SAPLPROX	002	TEST01
6	DIA	1752	Running		Yes			SAPLMG26	002	TEST01
7	DIA	1660	Running		Yes			SAPLTHFB	002	WIRTHENSOHN
8	UPD	2316	waiting		Yes					
9	UPD	2132	waiting		Yes					
10	UPD	2104	waiting		Yes					
11	UPD	516	waiting		Yes					
12	UPD	2188	waiting		Yes					
13	ENQ	2572	waiting		Yes					
14	BGD	2212	waiting		Yes					
15	BGD	1968	waiting		Yes					
16	BGD	1940	waiting		Yes					
17	BGD	1540	waiting		Yes					
18	SPO	1652	waiting		Yes					
19	UP2	1496	waiting		Yes					
20	UP2	2376	waiting		Yes					

- **Dialog mode with parallel processing:** In this scenario (**Figure 5**), *TEST01* uses one dialog process to handle the manual call of the outbound application. Because parallel processing works with asynchronous RFCs (aRFCs), which process work items in parallel, *TEST01* uses as many of the available dialog work processes (as specified in the server group settings) as it can to meet the needs of the outbound application.

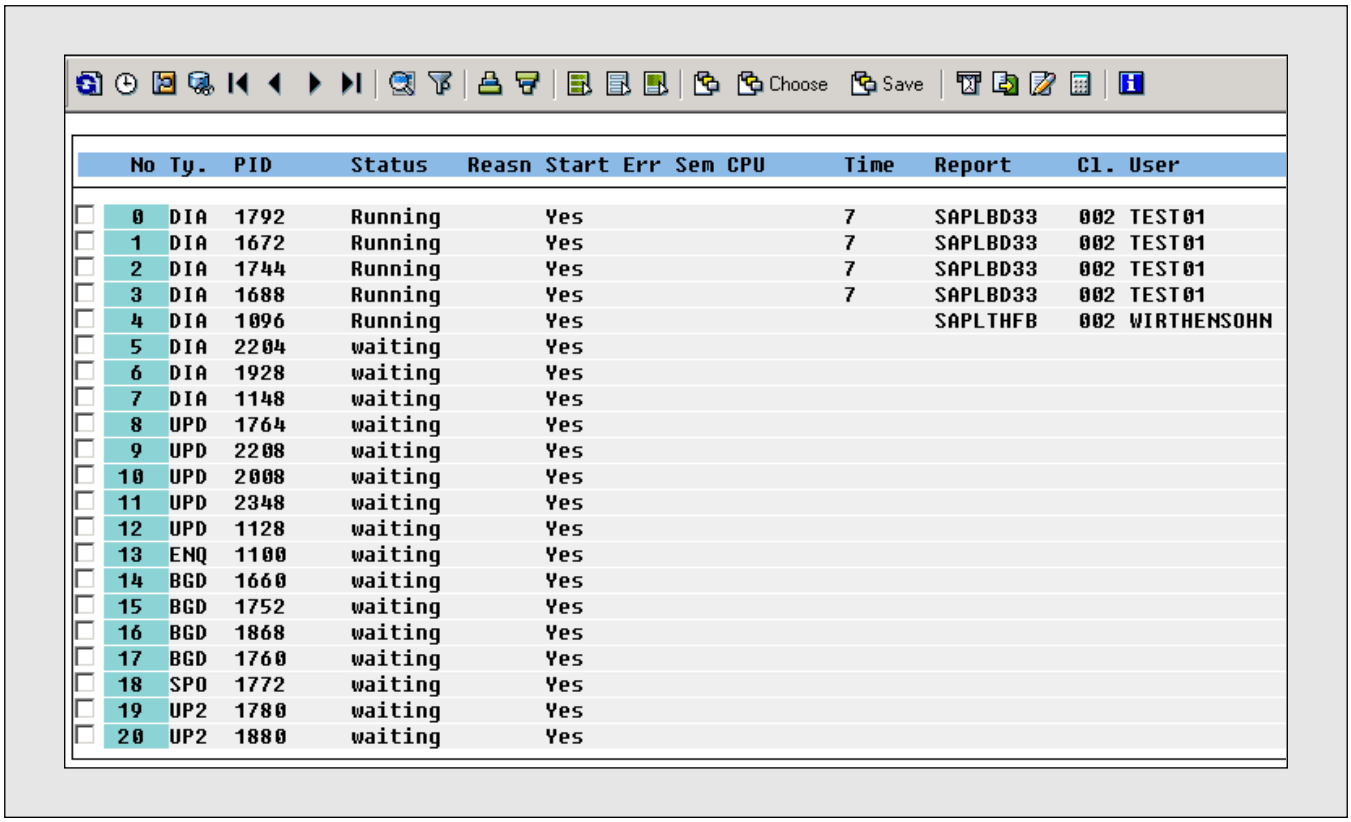
  Figure 6 shows the server group settings for the parallel processing performed in Figure 5. This server group was customized to occupy all available dialog work processes except three (by entering *3* in the *Min. no. of free WPs* field). Looking again at Figure 5, you can see that the system did not adhere to this restriction — all the dialog work processes are in use (one is occupied by user *WIRTHENSOHN*, and the rest are taken by user *TEST01*). At the very least, there should be one

Figure 6 Server Group Settings



 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

Figure 7 Creating an IDoc in Dialog Mode with Parallel Processing and Higher Restrictions



No	Ty.	PID	Status	Reasn	Start	Err	Sem	CPU	Time	Report	Cl.	User
0	DIA	1792	Running		Yes				7	SAPLBD33	002	TEST01
1	DIA	1672	Running		Yes				7	SAPLBD33	002	TEST01
2	DIA	1744	Running		Yes				7	SAPLBD33	002	TEST01
3	DIA	1688	Running		Yes				7	SAPLBD33	002	TEST01
4	DIA	1096	Running		Yes					SAPLTHFB	002	WIRTHENSOHN
5	DIA	2204	waiting		Yes							
6	DIA	1928	waiting		Yes							
7	DIA	1148	waiting		Yes							
8	UPD	1764	waiting		Yes							
9	UPD	2208	waiting		Yes							
10	UPD	2008	waiting		Yes							
11	UPD	2348	waiting		Yes							
12	UPD	1128	waiting		Yes							
13	ENQ	1100	waiting		Yes							
14	BGD	1660	waiting		Yes							
15	BGD	1752	waiting		Yes							
16	BGD	1868	waiting		Yes							
17	BGD	1760	waiting		Yes							
18	SPO	1772	waiting		Yes							
19	UP2	1780	waiting		Yes							
20	UP2	1880	waiting		Yes							

dialog work process that is not occupied by parallel processing — the monitoring transaction (i.e., user *WIRTHENSOHN*) should occupy one, and the outbound process (i.e., user *TEST01*) should occupy one, leaving one free — but *all* dialog processes are occupied, which means that user *TEST01* is occupying all remaining dialog processes. This is because not all RFC types follow instance profile and server group restrictions (remember from the sidebar on pages 6-7 that such restrictions apply only to aRFCs and tRFCs).[5]

Optimizing server group settings requires some trial and error — you'll want to try specifying some different numbers until you arrive at the desired result. In **Figure 7**, you can see what happens when the number of free dialog work processes in the server group is raised to five — the three dialog work processes that remain vacant can now be used as needed by dialog users, workflow items, and so on.

- **Background (batch) mode with parallel processing:** Like the previous scenario, here *TEST01* occupies one batch process to schedule the batch job, which calls the outbound application. *TEST01* then uses as many of the available dialog work processes (as specified in the server group settings) as it can to meet the needs of the outbound process (since the results are essentially the same as those for the previous scenario, the Process Overview screen is not shown for this example). As with the previous scenario, be

[5] This sounds like a serious drawback, but there are at least three good reasons to use a server group anyway: (1) parallel processing can significantly reduce the total processing time; (2) you can define the servers on which the processes will run; and (3) you can still restrict the number of aRFCs and tRFCs being used.

aware that RFCs other than aRFCs and tRFCs can occupy dialog work processes that, according to the server group settings, should remain free.

Parallel processing allows you to quickly create large amounts of IDocs by utilizing the dialog work process definitions in a server group. When defining a server group, keep in mind that because some RFCs are not aRFCs or tRFCs, the outbound process will not strictly adhere to the number of free dialog work processes you specify in the server group definition, so it is a good idea to specify a higher number of free dialog work processes than you actually need and then conduct some sample runs of the program to determine the optimal setting.

Optimizing Performance — The Communication Process

To transfer the IDoc, the communication process calls a remote function module that starts the IDoc inbound processing[6] on the receiver system and transmits the IDoc data as a parameter value.

Recall that the way an IDoc will be transferred in the communication process is specified through the port type (refer back to footnote 2 on page 5), and that for the purposes of this article, we are using tRFC as the port type and data transfer technology. Once the IDoc has been transferred to the port, status record 03 (Data passed to Port OK) is added to the IDoc's status record table. Keep in mind that status 03 does *not* indicate that the data transfer to the receiver system was successful, however! To verify that the tRFC's logical unit of work (LUW) was successfully committed — in other words, that the IDoc data was successfully transferred to the receiver system — you must run ABAP program RBDMOIND (Check IDoc Dispatch) after the data transfer, which will add status record 12 (Dispatch OK) to each successfully transferred IDoc. Running this ABAP program is not mandatory, but it is helpful if the dispatch information is needed for confirmation purposes (e.g., to monitor IDocs that haven't yet arrived in the receiver system).

Packet Processing in the Communication Process

When the outbound application sends IDocs immediately (when the outbound process is "coupled" to the communication process), there is one tRFC called for each IDoc. However, it is possible to send more than one IDoc per tRFC (in which case the outbound process is "decoupled" from the communication process) by customizing the packet size and sending the collected IDocs. **Figure 8** shows a graphical overview of packet processing.

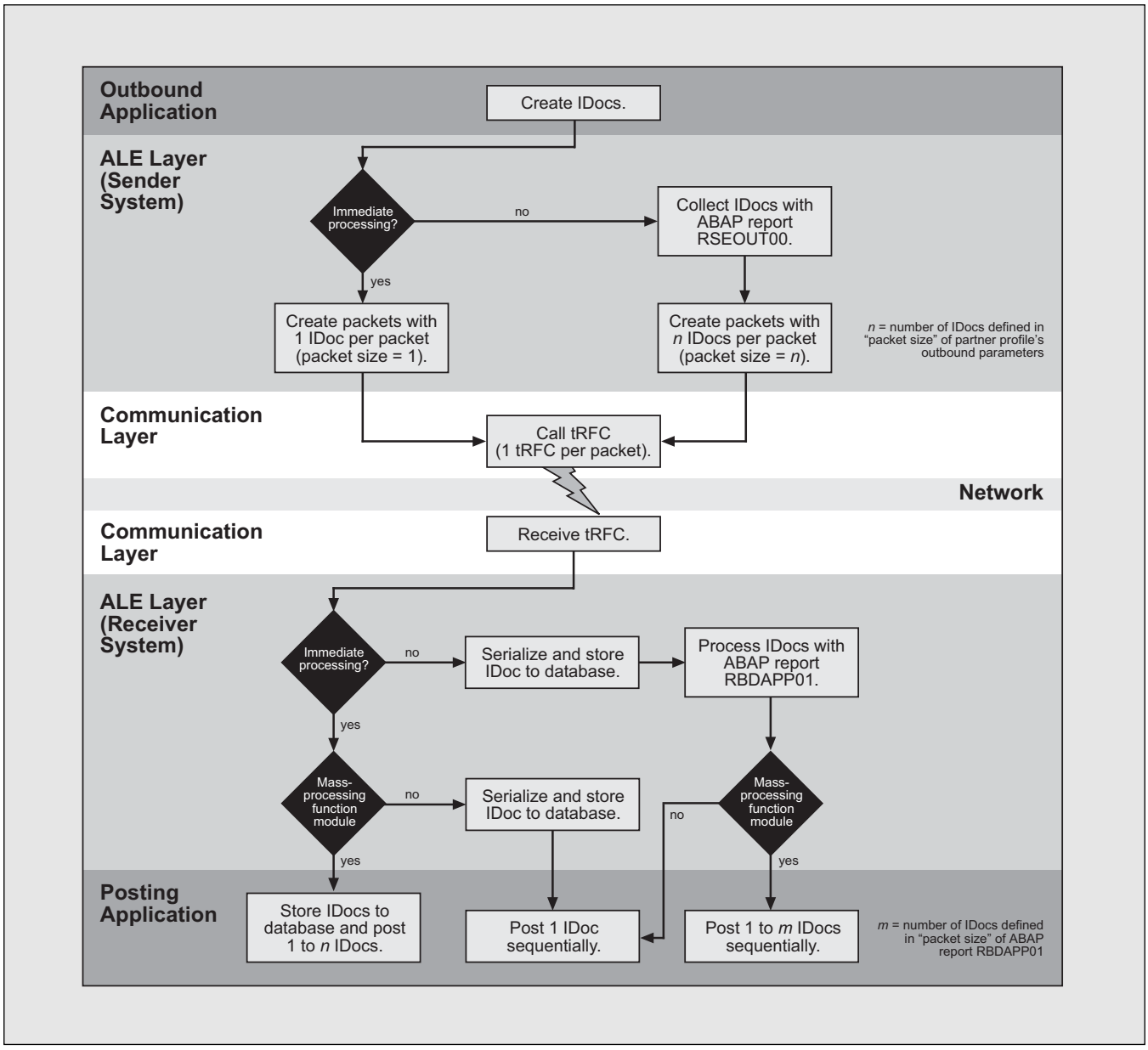
To maintain the packet size, in transaction *WE20* (Partner Profiles) you have to set the outbound parameter to *collect IDocs* and maintain the value of the *packet size* parameter. The value in the *packet size* field specifies the number of IDocs that will be included in one packet. When calculating a packet size, keep in mind that while the packet should carry as much workload as possible, it normally has a limit of 4,000 segments. Estimate the number of segments in your specific IDoc by going to the SAP Data Browser (transaction *SE16*) and selecting the IDoc type in table EDIDC (IDoc Control Records). Take a look at field *MAXSEGNR*, which displays the number of data segments in the IDoc. The number of IDoc segments to be transferred is the number of data segments plus one control segment.

If you use a customized packet size (i.e., larger than 1) and set the outbound parameter to *collect IDocs*, you need to run ABAP program RSEOUT00 (Process all selected IDocs) to send the IDocs. This program collects all of the selected IDocs, puts them

[6] As of SAP Basis Release 4.0, this is function module IDOC_INBOUND_ASYNCHRONOUS. In Release 3.x, this is function module INBOUND_IDOC_PROCESS. Port type tRFC's settings (located in transaction *WE21*), which specify how to connect to the receiver system, require a version definition. If the receiver system is running Release 3.1 (lower releases are not supported), the version must be *IDoc record types SAP Release 3.0/3.1*, which tells the communication process to call INBOUND_IDOC_PROCESS. If the receiver system is running Release 4.x or 6.x, the version must be *IDoc record types SAP Release 4.x*, in which case IDOC_INBOUND_ASYNCHRONOUS is called.

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

Figure 8 *Processing IDocs in Packets*



into a packet until the specified limit (the packet size defined in *WE20*) is reached, and transfers the packet to the receiver system through a tRFC. If the number of selected IDocs is larger than the specified packet size, RSEOUT00 will continue filling additional packets and sending them to the receiver system until no selected IDocs remain. If the number of selected IDocs is smaller than the specified size, RSEOUT00 will simply send the packet when there are no more IDocs left to add.

We will see the impact of packet size on the receiver side of ALE processing in the upcoming section on the inbound process.

> ✓ ***Recommendation***
>
> *Use appropriately sized packets whenever you can. You will reduce the administrative load on the sender system and the number of dialog processes consumed on both the sender and receiver side. Where "n" is the packet size (the number of IDocs in one packet), there will be n-times fewer tRFCs to handle.*

The Use of Work Processes in the Communication Process

If the communication process consumes too many dialog work processes, dialog users might find themselves blocked from using the sender or receiver system. Let's first look at how work processes are consumed in the communication process on the sender side and the receiver side, and then we'll look at some ways you can control the consumption of these processes.

- On the sender side, the communication process will occupy as many dialog work processes as required to connect to the receiver system via tRFC. Whether the communication process performs its tRFCs sequentially or in parallel depends on the outbound application and on the output mode — i.e., whether the outbound process is coupled to the communication process (send IDocs immediately) or decoupled from the communication process (collect IDocs and send later). For example:
  - If the outbound application is configured to send IDocs immediately *without* using parallel processing (i.e., one after the other), the communication process performs its tRFCs more or less sequentially (with one tRFC per IDoc, as explained in the previous section).
  - If the outbound application is configured to send IDocs immediately *with* parallel processing, the communication process also works in parallel.
  - If the outbound application is configured to send the IDocs via ABAP program RSEOUT00 (described in the previous section), RSEOUT00 will collect the IDocs sequentially but then allocate as many dialog work processes as needed and available to send the packet (or packets).
- On the receiver side, there is at least one dialog work process occupied per tRFC, and perhaps more if RFCs other than tRFCs are involved. In my experience, there are only two ways to counteract excessive dialog work process consumption in the receiving system:
  - Restrict dialog work process consumption by assigning quotas to RFCs at the system profile level (see the sidebar on pages 6-7 for more information on using this method).
  - Utilize the outbound queue of a queued RFC (qRFC) to specify the number of connections that may be used for sending tRFCs. This method enables you to restrict the number of parallel dialog work processes sent to a specific RFC destination (and thus the number of dialog work processes involved on the receiver side). The receiver system does not directly interact with the outbound queue of the sender system — it simply receives fewer simultaneous tRFCs, which in turn limits the number of dialog work processes needed for inbound processing.

Let's take a closer look at how to manage the outbound queue.

Configuring and Managing the Outbound Queue for Reduced Dialog Work Processes in the Receiving System

To configure the outbound queue:

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

1. Call transaction *SMQS* (qRFC Monitor), place the cursor on the destination you want, and click the *Registration* button.

> ✓ ***Tip***
>
> *If the destination you want is not listed in the SMQS overview screen, press the "Registration" button without selecting a destination. The resulting pop-up will allow you to enter a new destination.*

2. In the destination's pop-up window, go to field *MAXCONN* and adjust the default value for the maximum number of connections.

   For example, the default is usually 10 for an R/3 destination. You might change it to 2 in order to allow a maximum of two connections at the same time.

> ✓ ***Note!***
>
> *qRFCs have been available since Release 4.6A. Using transports, you can install the latest qRFC version on any SAP system running Release 3.x or later. If you don't know which version you have installed, call transaction SMQS and navigate to Information → Version. I would recommend using qRFC 6.20 at a minimum.*

In addition to configuring the outbound queue, you can assign a server group to the queue's outbound scheduler, which will allow you to determine the way in which dialog work processes on that group's servers are used for processing qRFCs. To assign a server group to the outbound scheduler, call transaction *SMQS*, navigate to *Edit → Change AS group*, and enter the name of an existing logon/server group. The system will not perform a validation check on your entry, so make sure that you type in the right server group name.

Finally, keep in mind that you might send IDocs to more than one receiver system at the same time, in which case the total number of dialog work processes involved in the communication process is approximately the sum of all maximum connections for each destination involved. This could become a performance problem in a central user management system that has many receiver systems. In such a scenario, it's best to send IDocs to the receiver systems in sequential order (to one receiver system after the other) by calling or scheduling a program that triggers the communication process for each receiver system in turn.

> ✓ ***Note!***
>
> *In the example here, we use the qRFC outbound queue to control the number of simultaneous connections. SAP also provides an inbound queue for qRFCs that can be used in addition to the outbound queue. For our purposes, we do not need the inbound queue. For more information on qRFCs, please refer to the SAP library under SAP NetWeaver Components → SAP Web Application Server → Middleware (BC-MID) → Remote Function Call (BC-MID-RFC) → Queued Remote Function Call (qRFC).*

Error Handling for tRFCs

And what happens if a transfer error occurs during the IDoc transmission? One of the attributes of a tRFC is an almost-guaranteed, one-time-only delivery. However, like a transactional LUW, a tRFC is committed *only* if all steps are performed successfully; otherwise, all steps are rolled back. While this behavior is fine for obtaining a defined transaction state, it is not sufficient to guarantee delivery of an IDoc, so SAP includes a mechanism that automatically repeats tRFCs with transaction errors. In a standard SAP system, each erroneous tRFC is scheduled as a background job to be called at a later point in time. This means that the background job relaunches the same tRFC. The naming convention for such a background job is *ARFC:tid*, where *tid* is the unique transaction ID of the original tRFC.

The number of connection attempts and the elapsed time between them are determined by the following settings, in the order listed:

- User exit SABP0003
- tRFC options in transaction *SM59* (RFC Destinations)
- Predefined settings from SAP

In user exit SABP0003, you can set the number of connection attempts and the time between them for a specified destination. In addition, you can use system fields, such as the one for user name (*SY-UNAME*), to specify further dependencies (e.g., specific connection attempts and a time interval for a specific user).

In transaction *SM59*, you can suppress connection attempt retries for a specified destination, or you can specify the number of connection attempts and the time between two attempts. To get to the tRFC settings in *SM59*, select the RFC destination you want and navigate to *Information → System Settings*. You will be able to set tRFC options for all RFC destinations except logical destinations or connections via ABAP drivers.

The SAP system's predefined settings for tRFCs are 30 connection attempts with 15-minute intervals between attempts. You can look up these settings when calling transaction *SM58* (Transactional RFC); confirm the initial selection screen to get to the screen that lists the tRFCs, and then navigate to *Information → System Settings* to view the predefined settings.

If a tRFC can't be performed successfully — and it will not be scheduled for reprocessing in a background job because of errors or retry limits — it remains in a tRFC database table as a tRFC with a transaction error and can be viewed with transaction *SM58*. tRFCs with transaction errors can be reprocessed with ABAP program RSARFCEX (Execute Calls Not Yet Executed).

For destinations with an unstable network connection to which many IDocs have to be sent, if the IDocs do not have to be transmitted in real time, you might want to suppress retries in transaction *SM59* and schedule ABAP program RSARFCEX or RSARFCCP (TRFC: Scheduler for Transactions with Status Connection Error) periodically — for example, every two hours.

> ✓ ***Tip***
>
> *When transferring mass data to a destination with an unstable network connection or reduced receiver system availability, it's possible that there will be numerous erroneous tRFCs repeatedly scheduled for batch processing, leading to reduced system availability for dialog users of the sender system. If all dialog work processes are repeatedly being used by these tRFCs, the result can be an unavailable sender system. If the exported data is not needed immediately in the receiver system, you can switch off retries for tRFCs and either schedule ABAP program RSARFCCP to run at a non-critical time or start it manually at a time you know the network and receiver system are available. To switch off retries, choose the desired RFC destination in transaction SM59, navigate to Destination → TRFC Options, and activate the "Suppress background job if conn. Error" field. If the data is needed immediately, you will need to try to improve the network connection.*

If you no longer need a tRFC with a transaction error, you can delete it using ABAP program RSARFC01 (tRFC Reorganization).

System Logon with ALE and the Load Distribution Feature

The tRFC connection's logon to the receiver system is another factor that can affect the performance of the receiver side's communication process as well as the

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

inbound process. A static logon to one server results in that server being heavily utilized while others are almost idle — a suboptimal distribution of the system load. There is a feature built into the SAP system that can help you avoid such scenarios: "load distribution."

To optimize the tRFC connection's logon to the receiver system, you can use the load distribution feature when configuring the RFC destination to that system. In this case, the feature will work like the load distribution of dialog users — where users log on to the system via a message server and are directed to a suitable application server — only distributed business processes, rather than users, are directed to the optimal application server (located on the receiver system). Load distribution can be very useful in an ALE distribution scenario. For instance, you might not want to run ALE processes on the database server, in order to avoid dialog work process shortages there, so you can choose a logon group with just application servers in it.

> ✓ ***Note!***
>
> *The load distribution feature works properly for RFCs as of Release 4.6C (with support package 42) and Release 4.6D (with support package 27). Do not use this feature in older SAP releases — in this case, the load distribution feature will apply its determination rules every 300 seconds. During this time interval, masses of RFCs might be directed to the server deemed most appropriate at the beginning of the interval, without taking into account the changed capacity situation. The result might be one overloaded application server and one or more idle servers.*

To turn on load distribution, select the *Load distrib.* button in transaction *SM59* and specify a message server. You will also be prompted for a logon group and system ID. To make a particular logon group available for use by a tRFC connection's load distribution function, you must call transaction *SMLG* (CCMS: Maintain Logon Groups) and activate the *Ext. RFC-enabled* field in the group-dependent attributes of that logon group; otherwise the logon group will be available only to dialog users.

Optimizing Performance — The Inbound Process

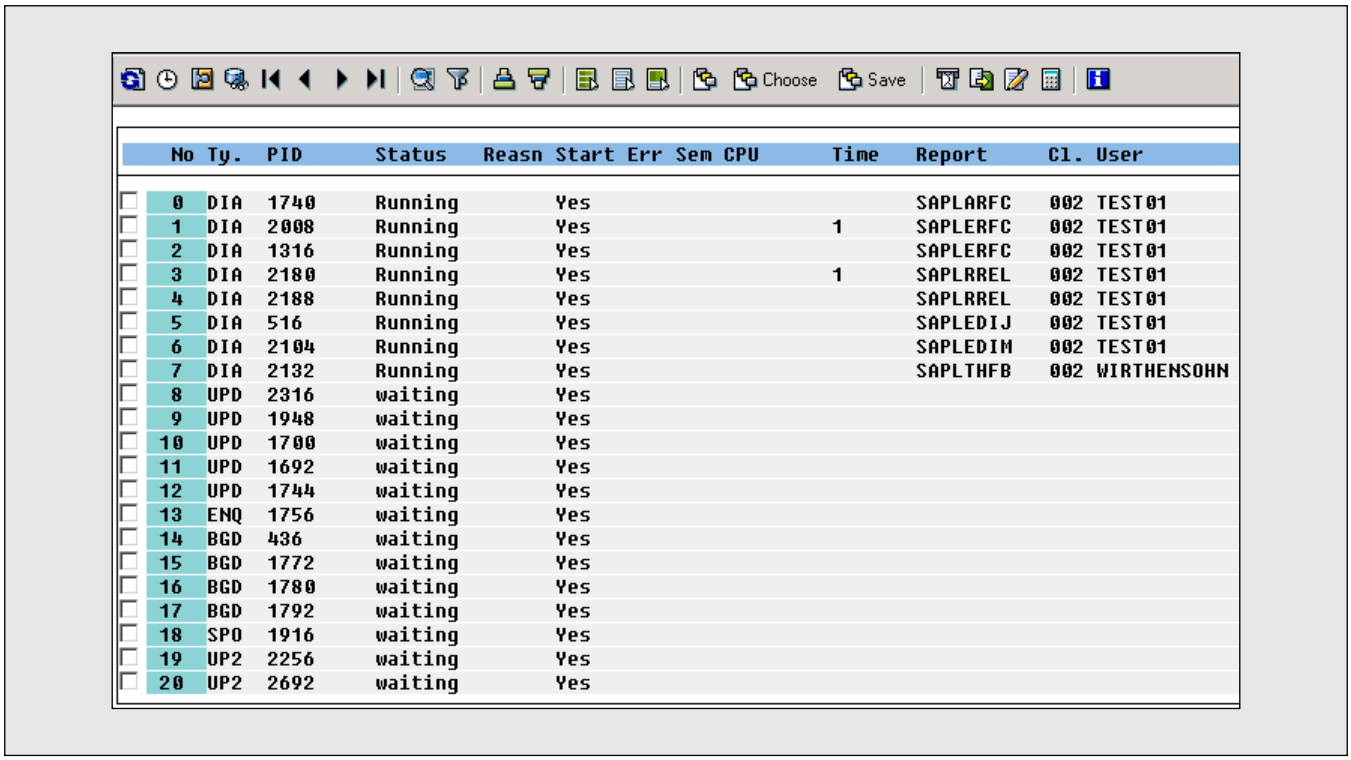
Like the outbound process, the inbound process takes place in two distinct steps (refer back to Figure 1):

1. The ALE layer receives the IDoc data from the communication IDoc, writes it to the IDoc database tables, and adds status record 50 (IDoc added) to the IDoc's status record table (EDIDS). The ALE layer then processes the IDoc by applying filters, conversion rules, and so forth, and adds status record 64 (IDoc ready to be transferred to the application) to table EDIDS.

2. The posting application reads the IDoc data, posts it as a business object, and adds either status record 53 to indicate that the posting was successful or status record 51 to indicate posting errors.

These two steps of the inbound process can be decoupled using parameter settings. Depending on the value of the inbound parameter *Processing by function module* in the partner profile (transaction *WE20*), IDocs are either processed immediately by the posting application (when the steps of the inbound process are coupled) or they are processed at a later time in the background or in dialog mode (when the steps are decoupled). When decoupled, the IDocs can be further processed by ABAP program RBDAPP01 (Inbound Processing of IDocs Ready for Transfer) or transaction *BD87* (Status Monitor for ALE Messages). For decoupled processing, the parameter option is *Trigger by background program*. This wording is a bit misleading, because RBDAPP01 can be started in dialog mode as well as in background mode. With coupled processing — parameter option *Trigger immediately* — the posting application will be triggered directly through the ALE layer.

Figure 9 Dialog Work Process Consumption in a Decoupled Inbound Process



No	Ty.	PID	Status	Reasn	Start	Err	Sem	CPU	Time	Report	Cl.	User
0	DIA	1740	Running		Yes					SAPLARFC	002	TEST01
1	DIA	2008	Running		Yes				1	SAPLERFC	002	TEST01
2	DIA	1316	Running		Yes					SAPLERFC	002	TEST01
3	DIA	2180	Running		Yes				1	SAPLRREL	002	TEST01
4	DIA	2188	Running		Yes					SAPLRREL	002	TEST01
5	DIA	516	Running		Yes					SAPLEDIJ	002	TEST01
6	DIA	2104	Running		Yes					SAPLEDIM	002	TEST01
7	DIA	2132	Running		Yes					SAPLTHFB	002	WIRTHENSOHN
8	UPD	2316	waiting		Yes							
9	UPD	1948	waiting		Yes							
10	UPD	1700	waiting		Yes							
11	UPD	1692	waiting		Yes							
12	UPD	1744	waiting		Yes							
13	ENQ	1756	waiting		Yes							
14	BGD	436	waiting		Yes							
15	BGD	1772	waiting		Yes							
16	BGD	1780	waiting		Yes							
17	BGD	1792	waiting		Yes							
18	SPO	1916	waiting		Yes							
19	UP2	2256	waiting		Yes							
20	UP2	2692	waiting		Yes							

> ✓ ***Note!***
>
> *Some applications — such as data synchronization applications used for data exchange between APO, CRM, BW, or R/3 — utilize qRFCs with the inbound queue. As part of the communication process, a qRFC used with an inbound queue writes the LUW to the database of the receiver system prior to the ALE layer processing. These queues can be activated through the QIN (inbound queue) scheduler using a program such as RSQIWKEX (Standard QIN Scheduler: Execution of Registered Inbound Queue). By activating the queues, the communication process will finish and the ALE layer will be started.*

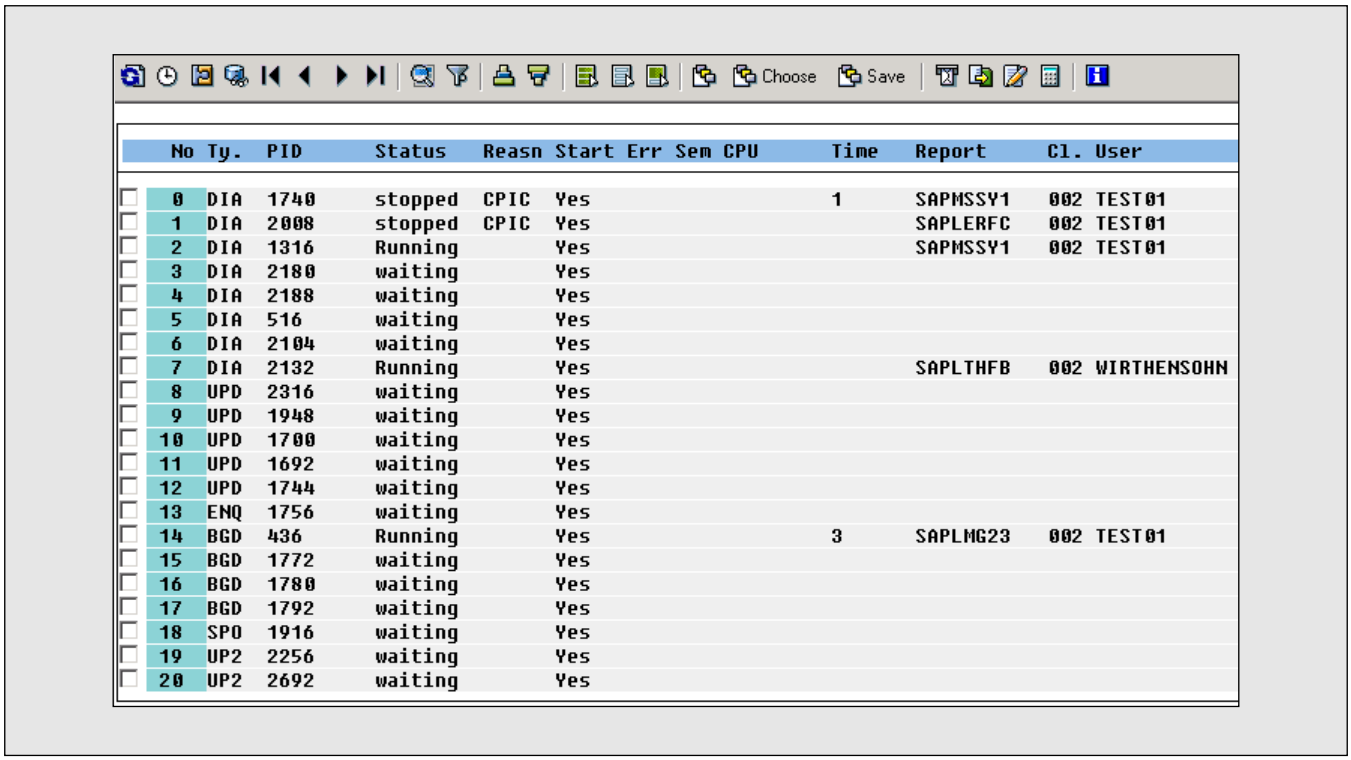
Dialog Work Processes in the Inbound Process

In the inbound process, the communication process triggers the ALE layer, which consumes at least one dialog work process for handling each packet and writing the data to the IDoc database tables in the receiving system. As we have seen, it is possible for many tRFCs to be executed at the same time in the communication process. When that happens, a huge number of simultaneous dialog work processes could be consumed in processing the received IDocs.

Using the Process Overview transaction (*SM50*), **Figure 9** shows the dialog work process consumption that occurs as the ALE layer receives many packets at the same time in a decoupled inbound process. As in the outbound process examples, user *WIRTHENSOHN* represents the monitoring application, which occupies one dialog work process. You will notice that all other dialog work processes are occupied by user *TEST01*, which represents the user of the tRFC (the ALE layer). The workload of a single processing task is comparatively small, but the impact on system availability can be serious because all available dialog work processes might be consumed repeatedly until all

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

Figure 10 RBDAPP01 Run in Batch Mode Without Parallel Processing



No	Ty.	PID	Status	Reasn	Start	Err	Sem	CPU	Time	Report	Cl.	User
0	DIA	1740	stopped	CPIC	Yes				1	SAPMSSY1	002	TEST01
1	DIA	2008	stopped	CPIC	Yes					SAPLERFC	002	TEST01
2	DIA	1316	Running		Yes					SAPMSSY1	002	TEST01
3	DIA	2180	waiting		Yes							
4	DIA	2188	waiting		Yes							
5	DIA	516	waiting		Yes							
6	DIA	2104	waiting		Yes							
7	DIA	2132	Running		Yes					SAPLTHFB	002	WIRTHENSOHN
8	UPD	2316	waiting		Yes							
9	UPD	1948	waiting		Yes							
10	UPD	1700	waiting		Yes							
11	UPD	1692	waiting		Yes							
12	UPD	1744	waiting		Yes							
13	ENQ	1756	waiting		Yes							
14	BGD	436	Running		Yes				3	SAPLMG23	002	TEST01
15	BGD	1772	waiting		Yes							
16	BGD	1780	waiting		Yes							
17	BGD	1792	waiting		Yes							
18	SPO	1916	waiting		Yes							
19	UP2	2256	waiting		Yes							
20	UP2	2692	waiting		Yes							

tRFCs are processed. Just think of the implications of a huge number of tRFCs continuously called in parallel from a sender system with only a few dialog work processes available on the receiver side to process them.

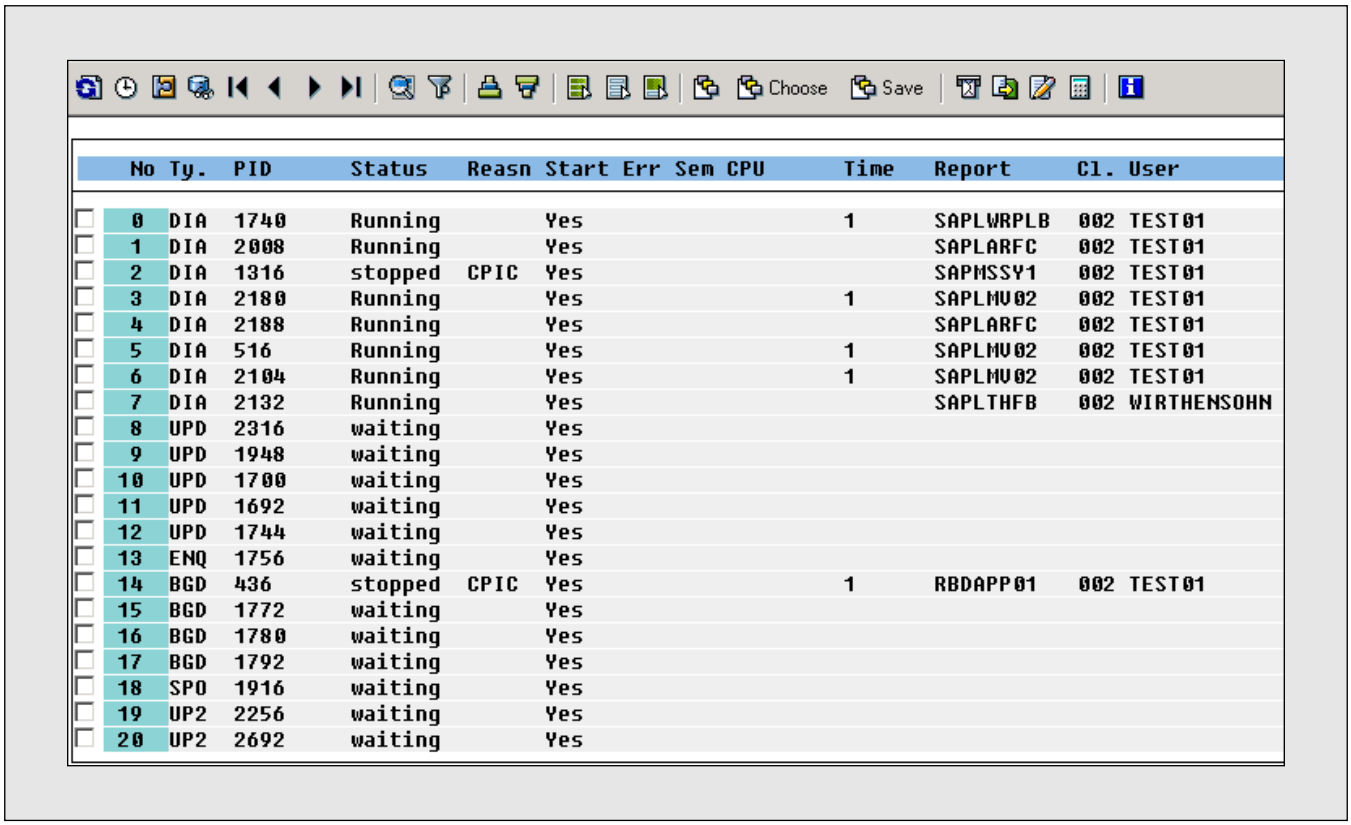
The workload of the posting application is considerably higher than the workload of the ALE layer processing. When the ALE layer processing is coupled to the posting application so that IDocs are posted immediately, there is almost no control over the workload because the ALE layer triggers the posting synchronously. The only way to gain control over the workload in this case is to decouple the inbound process so that the processing of the posting application can be scheduled at an appropriate time and performed separately using features like parallel or packet processing.

As of SAP Web Application Server 6.20, immediate processing of IDocs is no longer executed synchronously. A new process is started for the posting application only if there is a dialog work process available. Otherwise, as in a decoupled inbound process, the IDoc is assigned status record 64 (IDoc ready to be transferred to the application) with a description of the cause, so that you can determine the reason it hasn't been processed further. Afterward, the IDoc can be passed to the posting application using program RBDAPP01, which is normally used for posting IDocs in a decoupled inbound process.

RBDAPP01 supplies a parallel processing option; whether or not you choose to use this option has a significant impact on the number of work processes that are used for the posting process. For example:

- When RBDAPP01 is run *without* parallel processing, it posts IDocs sequentially. The packet size entered in the program's selection screen determines how many IDocs will be processed within one work process allocation. We will look at packet processing in the inbound process in more detail in the next section. **Figure 10** shows

Figure 11 RBDAPP01 Run in Batch Mode with Parallel Processing



No	Ty.	PID	Status	Reasn	Start	Err	Sem	CPU	Time	Report	Cl.	User
0	DIA	1740	Running		Yes				1	SAPLWRPLB	002	TEST01
1	DIA	2008	Running		Yes					SAPLARFC	002	TEST01
2	DIA	1316	stopped	CPIC	Yes					SAPMSSY1	002	TEST01
3	DIA	2180	Running		Yes				1	SAPLMV02	002	TEST01
4	DIA	2188	Running		Yes					SAPLARFC	002	TEST01
5	DIA	516	Running		Yes				1	SAPLMV02	002	TEST01
6	DIA	2104	Running		Yes				1	SAPLMV02	002	TEST01
7	DIA	2132	Running		Yes					SAPLTHFB	002	WIRTHENSOHN
8	UPD	2316	waiting		Yes							
9	UPD	1948	waiting		Yes							
10	UPD	1700	waiting		Yes							
11	UPD	1692	waiting		Yes							
12	UPD	1744	waiting		Yes							
13	ENQ	1756	waiting		Yes							
14	BGD	436	stopped	CPIC	Yes				1	RBDAPP01	002	TEST01
15	BGD	1772	waiting		Yes							
16	BGD	1780	waiting		Yes							
17	BGD	1792	waiting		Yes							
18	SPO	1916	waiting		Yes							
19	UP2	2256	waiting		Yes							
20	UP2	2692	waiting		Yes							

a work process overview of RBDAPP01 run in batch mode by user *TEST01* without parallel processing. As you can see, *TEST01* occupies one batch (background) process and three dialog processes.

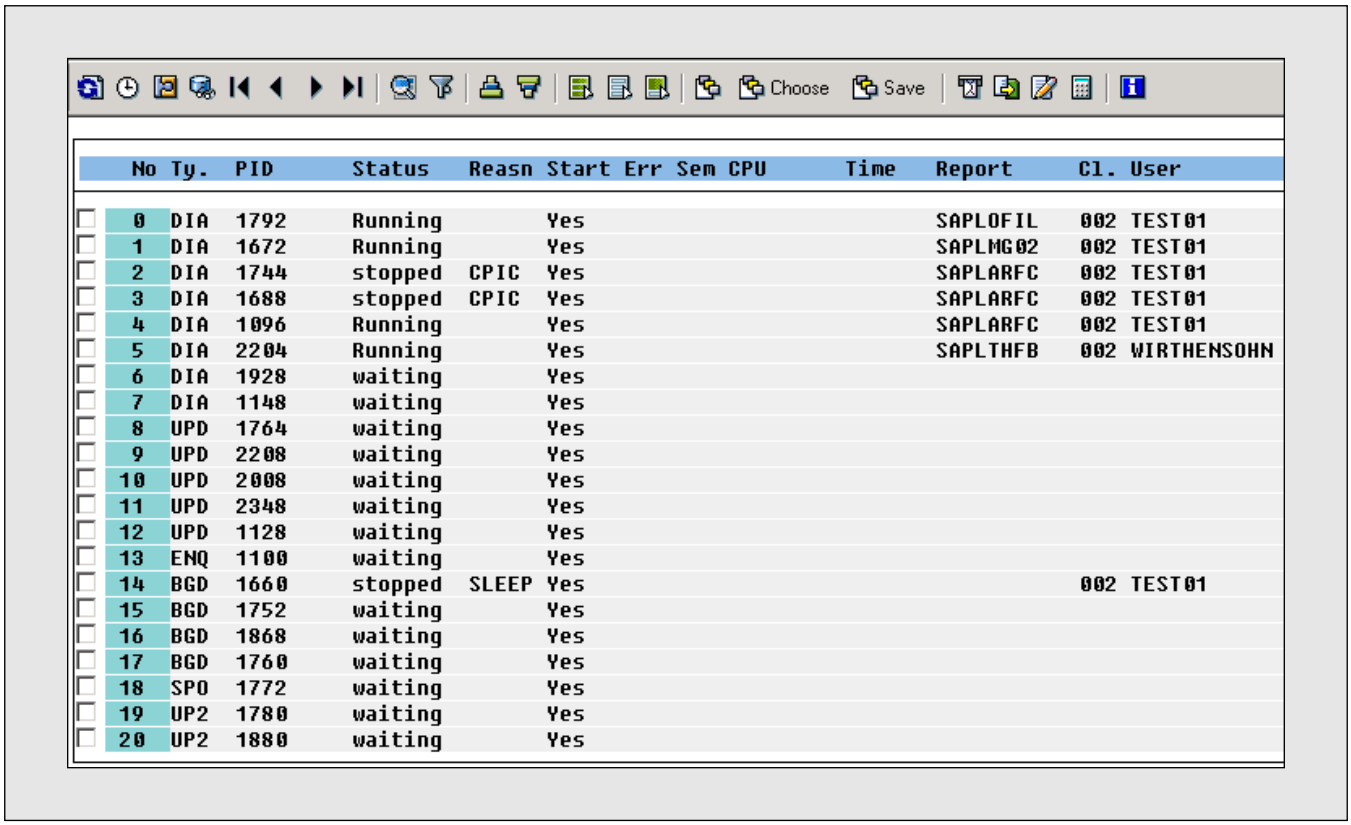
- Running RBDAPP01 in batch mode *with* parallel processing means that IDocs are posted in parallel packets. The available dialog work processes are defined by the settings of the specified server group. **Figure 11** shows a work process overview of RBDAPP01 run in batch mode by user *TEST01* with parallel processing. The batch process that started RBDAPP01 is occupied, just as it was in Figure 10 when parallel processing was not used, but in this case all of the available dialog work processes are occupied as well — even though I used the same server group settings that I used for the earlier outbound process examples (which specified a minimum of three free dialog work processes).

  As you'll recall from the earlier outbound process examples, these restrictions are only valid for aRFCs (which are used in parallel processing) and tRFCs; other types of RFCs, which may be called out of the main receiving program, do not adhere to these restrictions.[7] For this reason, the process will not strictly adhere to the number of free dialog work processes you specify in the server group definition, so it's a good idea to specify a higher number than you need and do some trial runs to determine the number that works best. **Figure 12** uses the same scenario used in Figure 11, only the server group's minimum number of free dialog work processes has been raised from three to five. As a result, two dialog work

[7] Remember that despite this limitation, it is still a good idea to use server groups (refer back to footnote 5 on page 13).

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

Figure 12 RBDAPP01 Run in Batch Mode with Parallel Processing and Higher Restrictions



No	Ty.	PID	Status	Reasn	Start	Err	Sem	CPU	Time	Report	Cl.	User
0	DIA	1792	Running		Yes					SAPLOFIL	002	TEST01
1	DIA	1672	Running		Yes					SAPLMG02	002	TEST01
2	DIA	1744	stopped	CPIC	Yes					SAPLARFC	002	TEST01
3	DIA	1688	stopped	CPIC	Yes					SAPLARFC	002	TEST01
4	DIA	1096	Running		Yes					SAPLARFC	002	TEST01
5	DIA	2204	Running		Yes					SAPLTHFB	002	WIRTHENSOHN
6	DIA	1928	waiting		Yes							
7	DIA	1148	waiting		Yes							
8	UPD	1764	waiting		Yes							
9	UPD	2208	waiting		Yes							
10	UPD	2008	waiting		Yes							
11	UPD	2348	waiting		Yes							
12	UPD	1128	waiting		Yes							
13	ENQ	1100	waiting		Yes							
14	BGD	1660	stopped	SLEEP	Yes						002	TEST01
15	BGD	1752	waiting		Yes							
16	BGD	1868	waiting		Yes							
17	BGD	1760	waiting		Yes							
18	SPO	1772	waiting		Yes							
19	UP2	1780	waiting		Yes							
20	UP2	1880	waiting		Yes							

processes have been left vacant for use as needed. As you can see from the examples shown here, running RBDAPP01 without parallel processing does not consume an excessive amount of dialog work processes, but processing time might be long if many IDocs have to be posted. And although running RBDAPP01 with parallel processing can really improve the processing time, you'll need to verify the server group specifications by observing the dialog work process usage when the program is running. Adjust your settings according to your observations until you reach the number of vacant dialog work processes you need.

Also, it is a good idea to decouple the steps of the inbound process whenever possible so that IDoc processing can take place in the background. This will make your system more stable — providing you with more control over the processing and ultimately better performance for all dialog process consumers — because the processing that requires most of the system resources (the posting of the data) is controlled by the receiver system. The more information to be exchanged, the more important it is to decouple the processing in the ALE layer from the processing in the posting application — so that you can process time-critical business data (e.g., business documents) immediately and save the processing of "non-time-critical" data (e.g., master data) for later, for example.

If you choose to process time-critical data immediately and non-time-critical data later, keep in mind that a time-critical business document might be based on non-time-critical master data. So, if you have decoupled master data processing, it is possible that even though it was transferred earlier, the data required by your business document won't be posted because RBDAPP01 (the posting program) hasn't run yet, in which case the IDoc of the business document will receive an error status.

Figure 13 Packet Processing of IDocs in the Inbound Process

Immediate Processing?	Mass-Processing Function Module?	How IDoc Inbound Processing Will Be Performed
Yes	Yes	As many IDocs will be posted within one call as are delivered in the packet through the communication process. Recall that the number of IDocs within one packet is determined through the sender system's outbound parameters *output mode* and *packet size*.
Yes	No	All IDocs from the packet will be serialized and posted by the IDoc inbound function module one after the other.
No	Yes	All IDocs from the packet will be serialized, processed by the ALE layer (filters, data conversions, and so forth are applied), and stored to the database. The *packet size* parameter in program RBDAPP01 determines how many IDocs will be posted by the IDoc inbound function module within one call.
No	No	All IDocs from the packet will be serialized, processed by the ALE layer (filters, data conversions, and so forth are applied), and stored to the database. For the IDocs to be posted by the IDoc inbound function module one after the other, run program RBDAPP01.

> ✓ ***Tip***
>
> *To address the problem of posting business documents that are dependent on non-posted data, you can, for instance, reprocess the IDoc of the business document with transaction BD87 as soon as the required data is posted. In some cases, you can also solve these kinds of problems by implementing ALE's "serialization" feature, which allows you to define interdependent messages. If you choose to use this option, be sure to examine the SAP Notes for serialization-related performance issues.*

Packet Processing in the Inbound Process

There are two complementary parameters that control packet processing in the inbound process (for a reminder of how packets are processed, refer back to Figure 8 on page 15):

- The inbound parameter *Processing by function module*, which is specified in transaction *WE20* (Partner Profiles), determines the coupling or decoupling of the inbound process.
- The parameter *Input typ*, which is specified in transaction *BD51* (Characteristics of Inbound Function Modules) under the IDoc inbound function module attributes, determines the posting

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.

Optimization Recommendations in a Nutshell

- ☑ Use parallel processing with server groups to achieve better processing throughput.
- ☑ Group IDocs in packets whenever possible to achieve better processing performance.
- ☑ Decouple the outbound process from the communication process and decouple the inbound ALE layer from the posting application to achieve better control over the distributed business process, higher processing security, and better availability for dialog users.
- ☑ If the receiver system provides logon groups for RFC communication, use one to connect to the receiver system so that the processing is routed to the best available application sever.

application's "mass-processing" capability. A value of *0* in the *Input typ* parameter indicates that a function module is capable of processing more than one IDoc within one function call — i.e., it is a mass-processing function module.

Depending on how these parameters are set, there are four possible scenarios for packet processing in the inbound process, as outlined in **Figure 13**.

As in the outbound process, use a reasonable packet size (number of IDocs) whenever you can to reduce both the administrative load and the database load.

✓ ***Note!***

When using packets and parallel processing with RBDAPP01, keep in mind that the packets, not the IDocs, are processed in parallel. If there are not enough packets to process in parallel, you will not gain any of the advantages of parallel processing. For example, if you have 60 IDocs to post and you set the packet size to 100, you will have just one packet with 60 IDocs — and no parallel processing.

Setting the packet size in RBDAPP01 to a value larger than 1 generally reduces the administrative load on the system. This is true even if the function module of the posting application is not capable of mass processing — a work process has to be allocated each time a packet of IDocs should be processed, and the fewer packets that have to be processed, the fewer work process allocations the system has to perform.

Conclusion

ALE is not the only part of the SAP system that requires system resources. Though it may seem obvious, the importance of this fact cannot be overstated. Optimizing ALE means optimizing ALE processes while leaving enough system resources for other users (dialog users, SAP Business Workflow users, Internet Communication Manager processes, and so forth). By following the recommendations in this article, you can choose the best possible processing options for efficient use of your own system resources, gain as much control over your processes as possible to maintain system stability and availability, and find the right balance of system resources between ALE and other users for optimal system use.

Additional Resources

- ☑ Search the SAP Service Marketplace (**http://service.sap.com**) for SAP Notes related to ALE performance. The Service Marketplace contains a wealth of information about different performance aspects within ALE (e.g., performance issues involved with various outbound applications and posting applications, along with performance issues regarding serialization and large or fast-growing tables in the communication process or in the ALE layer).
- ☑ Refer to the SAP library's *Optimizing ALE Performance* documentation under *SAP NetWeaver 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*. This documentation contains brief information on some of the performance aspects mentioned in this article along with some valuable tips for IDoc archiving and reorganization.

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.

 ©2003 SAP Professional Journal. Reproduction prohibited. All rights reserved.