Leveraging the R/3 Warehouse Management Structure with the MM-MOB and WM-LSR Interfaces

Michael Ottenstein

Michael Ottenstein studied at the University of Mannheim, Germany. He joined SAP AG in 1996 as a member of the Integration & Certification Center, where he gained experience with SAP's communication technologies. Since 1997, he has been responsible for the certification of SAP's interfaces in the areas of MM, WM, and SD.

(complete bio appears on page 84)

Use of the R/3 Warehouse Management (WM) module is based on a simple supposition. You use it to optimize material flow. You use it to place stock in the storage locations where it makes the most sense in terms of space and availability. WM is the module that oversees management of complex warehouse structures that are characterized by diverse types of storage areas, right down to the storage bin level. This is the place where the R/3 system processes stock movements such as goods receipts, goods issues, and stock transfers using predefined stock placement and stock removal strategies.

Integrating this R/3 module with the numerous mobile data entry devices and external warehouse management systems that typically drive these kinds of stock movements is made possible by two interfaces — Mobile Data Entry (MM-MOB) and Warehouse Control Unit (WM-LSR).

The MM-MOB interface enables mobile entry and transfer of data to and from SAP. This interface is actually "module-neutral," meaning that it can communicate information from bar-code and weighing instrument systems not only to the R/3 Warehouse Management (WM) module, but to the Inventory Management (IM) and Sales and Distribution (SD) modules as well. If, for example, you need to post goods receipts from an external system to the IM system, issue a stock placement or stock removal request, or send weight information from scales into IM, you would do so through this interface. If you want to transfer goods from production to the WM system, it is also through this interface. Interfacing the Sales and Distribution system to picking systems or sending packing data to SD is done via this interface as well.

The WM-LSR interface enables the sending and receiving of information between external warehouse management systems (automated storage and retrieval systems, fork-lift control systems, picking systems, carousels, etc.) and the warehouse control units in the SAP R/3 environment. Placement of manufactured goods into stock from the warehouse, for example, takes place via this interface, as does placing goods into stock in a manually allocated storage bin with a system message. Creation of a replenishment transfer order for the production plant and mobile entry of count data for system inventory records are done via this interface as well.

These two interfaces open up a vast array of application scenarios along the lines of those presented in **Figure 1** and **Figure 2**.

Figure 1 Common Mobile Data Entry (MM-MOB) Application Scenarios

Posting goods receipts from an external system to the Inventory Management (IM) system. A generic procedure might have the order number on the delivery note read in with a bar-code scanner, downloaded from R/3 to the external system, and displayed on the handheld device. The items of the delivery note are entered in the external system, which checks the order against the delivery, and sends the data. The goods receipt is posted in R/3.

Stock placement from the production plant to the Warehouse Management (WM) system. Placing manufactured goods into stock involves the confirmation of pre-planned transfer orders. These transfer orders must have been created in the R/3 system beforehand, then printed and sent to the production plant. The pallets that are sent to the warehouse from the production plant should be accompanied by a transfer order document. The transfer order number is scanned with the handheld device. The external system generates the necessary data for the confirmation of the transfer order. The external system sends the data to R/3, where it is posted in the WM system.

Stock placement to the WM system with manual storage bin allocation. Goods are simply placed in a storage bin. The storage bin, material, and quantity are scanned in (or keyed in manually). The external system sends the data to R/3, where a transfer order is generated. As the physical movement has already taken place, the generated transfer order will be confirmed immediately.

Entering inventory count data with warehouse management. As soon as warehouse management systems are used, it is necessary to draw up inventories in accordance with the storage bins. Here, a developer would create inventory records in the WM system, send those records from R/3 to the external system, enter inventory count data on the handheld devices, and send count data to R/3 and post it in the WM system.

Interfacing with picking systems. For each shipping point, you can specify whether the data in a picking list is to be output in paper form or sent to an external system. Delivery header and item data that is relevant for picking is sent. In both cases, a picking request is generated. Picking requests that have been sent to an external system can be updated via an IDoc. It is possible to update picked quantities, report batch splits, adjust the delivery quantity to the picked quantity, and to initiate the posting of a goods issue.

Report packing data to the Sales and Distribution (SD) system. It is possible to update the current packing data for a delivery document in different ways. Packed items can be reported as well as packed shipping elements. Delivery items can be created from the shipping elements so that these can be billed and be processed by the Inventory Management system.

Connection of scales. In this type of MM-MOB application scenario, the data is not entered on a mobile device. A typical customer requirement is to weigh a truck when it is empty, load it, weigh it again, calculate the difference, and send the data to R/3. This results in the posting of a goods receipt.

Figure 2 Common Warehouse Control Unit (WM-LSR) Application Scenarios

Manual warehouse. Here, the WM system assumes full control over the management of the warehouse and also takes care of inventory. WM manages the storage bins and the material stocks, and generates all possible goods movements (stock placement/removal/transfer and posting changes). The storage bins for movements is determined with defined stock placement and removal strategies.

Semi-automatic warehouse. Here, the WM system assumes full control over the management of the warehouse, managing storage bins and material stocks, generation of all possible goods movements, stock placement/removal strategies, and physical inventory. The external system merely performs the warehouse control functions. It has the control of the manual and automated conveyors, it controls the material flow, and it can do an optimization of resources.

Fully automatic warehouse. Here, both the warehouse as a whole and the specifications of automatic warehouse systems must be taken into account in order to determine how the functions can be distributed effectively between the WM system and an external system. The WM system manages the storage bins and material stocks, generates all possible goods movements, and determines the storage bins for movements with the defined stock placement/removal strategies. The external system has control over the conveyors and the material flow and does an optimization of resources. Unlike the scenario for semi-automatic warehouses, all stock placements take place via an Identification Point, and all stock removals take place via a Pick Point.

Black Box. Imagine a warehouse complex where different storage techniques can be used in the individual warehouses. For example, it would be possible to have a warehouse with several manual storage types and one or more automatic storage types. The automatic storage type is highly automated and very complex regarding its structure and storage technique. In order to define a goods movement, information on the current status of the warehouse systems is required. A large number of communication processes between the WM system and the external system would have to take place in this case. Certain events in the warehouse control unit would require an immediate response from the warehouse management system — which is not possible with the asynchronous interface currently used. The WM system can be used for the entire warehouse complex. It distributes the individual stock placement and removal procedures amongst the individual storage types. In manual warehouses, all aspects of warehouse management are controlled by the WM system. The automatic warehouse is regarded as a black box — i.e., the WM system does not recognize any storage bin stock. In this case, the WM system would only know a summarized stock for each material. R/3 would still create the transfer orders for goods movements; all other tasks would be performed by the external system.

Interfacing the R/3 sytem to an external Warehouse Management system. When a WM system is used, communication takes place between the SAP WM system and the external system. An interface between Materials Management (MM), Sales and Distribution (SD), or Production Planning (PP) is not necessary, since it is always the WM system that is activated first. There are, however, warehouses for which the use of an external warehouse management system would be appropriate. With R/3 Releases 3.x and 4.0, there is no standard interface for connecting the individual components MM, SD, and PP to an external warehouse management system. The task of interfacing is left to the customer. There are a number of standard SAP objects that can be used here — e.g., the standard IDoc WMMBID01 and the user exit MB_CF001 for interfacing the IM module. As of R/3 4.5A it will be possible to link a distribution center as an independent component with the R/3 system or to link the WM system (as a distribution center) with any one ERP system. In both cases, the ERP system and the WM system work on different computers.

As developers, the things you need to understand in order to implement an MM-MOB- or WM-LSR-based application scenario are these:

- The **terminology** used to characterize the warehouse structure in R/3, which may be different from that used to characterize the warehouse structure of your target external system.
- The specific **functions** that are provided by the MM-MOB and WM-LSR interfaces.
- The **IDocs and message types** you must invoke in order to enable your application.
- The **steps you need to take** in order to customize the MM-MOB and WM-LSR interfaces.

These are the topics I will cover in this article. Once you understand these facets of the MM-MOB and WM-LSR interfaces, you should have an easier time creating applications that integrate mobile data entry devices and warehouse management systems with the R/3 warehouse management structure.

As developers, in order to implement an MM-MOB- or WM-LSR-based application scenario, you need to understand the terminology used to characterize the warehouse structure in R/3, the specific functions that are provided, the IDocs and message types you must invoke, and the steps you need to take to customize the interfaces.

R/3 Warehouse Terminology

One or more elements of the Warehouse Management structure will be in all the IDocs that are exchanged between your application and the external device or system. So, if you are going to build applications that integrate with the R/3 Warehouse Management (WM) module, you must understand the hierarchical WM structure and the terminology used to characterize those structures. Otherwise, you can run into trouble. A "storage location," for example, can mean different things inside and outside R/3. Some warehouse management systems recognize a warehouse number as a "storage location." "High rack storage" is another term that can have different meanings. Within R/3, "high rack storage" is a storage type. In another system, it can denote an R/3 warehouse number. When building an application, if you don't understand the precise meaning of the R/3 WM terms, you can't fill in the IDocs correctly, and you will find yourself facing error messages like "Storage location 123 not defined."

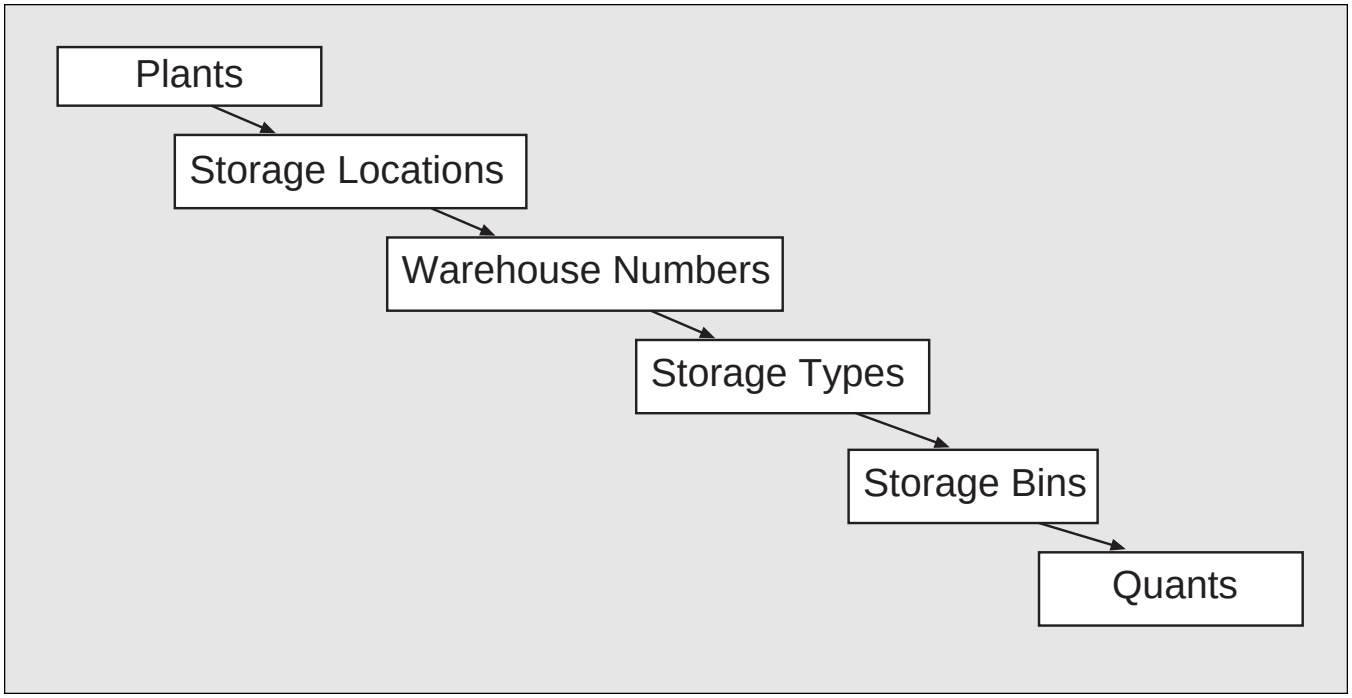
Within R/3, the entire warehouse structure is managed under a single *warehouse number* (also referred to as *warehouse complex*), as shown in **Figure 3**. This is different than the way things are typically done in other systems.

Note: Plants and storage locations are present in the system, even when the WM module is not used, whereas warehouse numbers and everything below exist only when the WM module is in use.

All the warehouses that are associated with a particular plant are grouped together into *storage locations*. Within a warehouse number, the various storage areas are defined as *storage types*. Again, this is different than the way it is done in many other systems. High rack storage, block storage, and fixed bin storage are all examples of storage types. Storage types are differentiated according to technical and organizational characteristics, such as goods receipt area, goods issue area, high rack storage area with random picking, block storage area with rows of the same material, and picking area with fixed storage bins.

Within each *storage type* the individual *storage bins* are defined. A storage bin is the smallest geographical or organizational unit (also called coordi-

Figure 3 *The Warehouse Structure in the R/3 System*



nate) that can be addressed by the R/3 system. Storage bins are identified by a bin type (e.g., bin type P1 for high bins) and are organizationally grouped into storage sections (e.g., fast-moving items and slow-moving items).

The existence of a material in a bin is defined as a *quant*. Quants are identifed by material number, batch number (if the material is batch managed), stock category, special stock, and plant and storage unit number (if SU management is activated for the respective warehouse number and storage type). As far as I know, there is nothing like a quant in other systems.

In storage bins designated for single-material storage, only one quant may be stored. In bins designated for mixed storage, different materials, or quants, may be stored in a single storage bin. A specific type of palletization, or container, for a quant is designated by the storage unit type. Only storage units of a particular storage unit type are allowed to be stored in a particular type of storage bin.

Once you understand the R/3 warehouse structure, you should turn your attention to the warehouse structure of the external system to which you want your application to interface. Suppose, for example, that you decide that the external system will be responsible for storage bin management. You would then need to know the hierarchical structure of that external system, where it might be the case that the identifier for the storage bin is unique. There may be more than one storage type that has Aisle 01-Stack 02 – Level 03 and thus the coordinates 01-02-03.

Functions of the MM-MOB and WM-LSR Interfaces

The MM-MOB and WM-LSR interfaces enable you to integrate the R/3 Warehouse Management module with a broad range of mobile entry and

Figure 4 MM-MOB Functions

Function	Example
Entry of goods movements	Inform R/3 about the receipt of a material for purchase order 5432.
Entry of inventory count data	Report the quantity of material "ABC" found at storage bin 02-03-04.
Reporting of packing data	Tell R/3 that you have packed delivery number 4711, used two cartons and a pallet, and perhaps report upon which material is located in which carton.
Transfer of delivery notes to an external system	Instead of a picking list in paper form, send the information via an IDoc.
Reporting picking quantities for delivery documents	Report that you have picked 30 pieces of material "hammer" for delivery number 4711, item number 10.
Entry of movements in the warehouse	This entry appears under mobile data entry functions, because it is possible to execute the functions of the WM-LSR interface with radio frequency devices.

external warehouse management systems. But before you can create these types of applications, you've got to understand what functions these interfaces are capable of supporting, and if any of those functions can be supported by the system to which your application will be interfacing. Only then can you decide whether or not your application should be responsible for a specific activity, or whether the external system should shoulder that responsibility. (Management of storage bins, for example, is an activity that can be performed by R/3 or by an external warehouse management system.)

The functions that are supported by the MM-MOB interface are summarized in **Figure 4**. The functions that are supported by the WM-LSR interface are summarized in **Figure 5**. At the top of the figure, you'll find functions associated with sending information to external systems from R/3. The bottom half of the figure lists the functions that are associated with receiving data from an external system.

Consider two alternative application scenarios for the management of high rack storage, whereby:

- A container of goods arrives
- The goods are scanned with a handheld device
- The goods are placed on pallets (if necessary) and sent via conveyor into the warehouse
- The pallets are identified and put away into high rack storage

In scenario #1 (shown in **Figure 6**), SAP controls high rack storage. When the container arrives:

1. The purchase order (PO) number is scanned by the handheld device, and the goods movement (GM) is sent to SAP R/3. To set this in motion, you employ IDoc WMMBID01, message type WMMBXY for goods movement.

2. SAP automatically posts the transfer order (TO), and then sends that TO back to the subsystem. The appropriate IDoc type is WMTOID01, message type WMTORD.

3. The pallet is identified at the ID point, and the TO is confirmed and put away.

Figure 5 WM-LSR Functions

	Function	Example
Sending Data	Reporting of transfer orders	Pick up pallet at the goods receipt area, and bring it into high rack storage.
	Reporting of multiple process reference numbers	There are five transfer orders for the same releases, with customer, but this customer does not accept partial deliveries.
	Cancellation requests for transfer orders	The customer has cancelled an order, so you don't need the material, which means, if it was a large order, you can probably reschedule your resources in the warehouse.
	Sending an inventory bin location list	Tell me how much of material "ABC" is located in storage bin 02-03-04.
Receiving Data	Generation of reported transfer orders	If the external system is responsible for stock placement strategies and has executed a movement, it needs to inform R/3 about it.
	Confirmation of transfer orders	Confirm that the physical movement has been executed, and that the pallet has reached its destination.
	Cancellation of transfer orders that have not been confirmed	The conveyor for one of the aisles is broken, and you cannot execute a transfer order.
	Movement of storage units	Each pallet is identified by a number. You have moved a pallet, and you want to report this to R/3.
	Blocking and unblocking of storage bins	Inventory counting takes place in two aisles. You do not want any stock placement and/or removal during this time, and you block the storage bins. Later the bins will be unblocked again.
	Generation of transfer requirements	A transfer requirement is a planned movement, not the physical execution. Sending transfer requirements results in the creation of transfer orders at a later point in time within R/3.
	Reporting of inventory count data	Report the quantity of material "ABC" found at storage bin 02-03-04.

Figure 6 Scenario #1: SAP Controls High Rack Storage

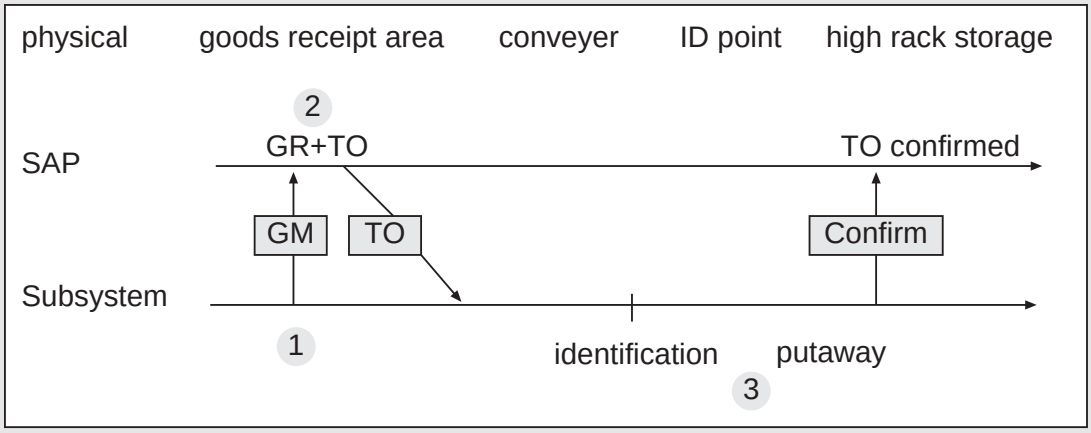
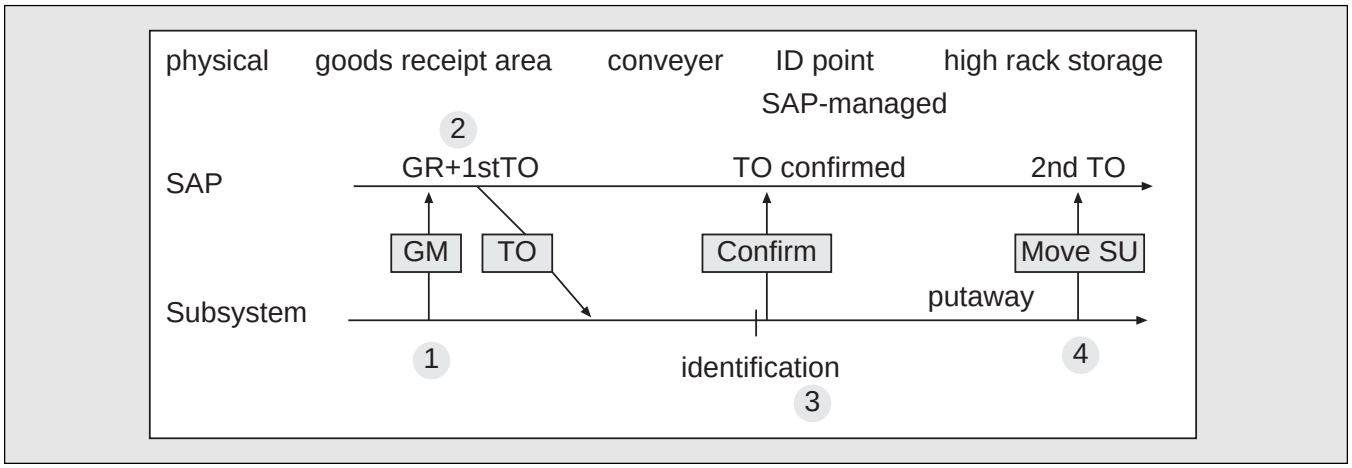


Figure 7 Scenario #2: Subsystem Controls High Rack Storage



In scenario #2 (shown in **Figure 7**), a subsystem controls high rack storage. When the container arrives:

1. As is the case with scenario #1, the purchase order (PO) number is scanned by the handheld device and the goods movement (GM) is sent to SAP R/3.

2. Just like scenario #1, SAP posts the transfer order (TO) automatically, and sends the TO back to the subsystem (IDoc type WMTOID01, message type WMTORD), but SAP does not establish the placement strategy for the goods. This is left for the external subsystem.

3. The pallet is identified at the ID point. Since the subsystem (not SAP) determines the storage bin where the pallet should be brought to, the transfer order is confirmed and put away when the pallet is at the identification point (step 3). Here, we employ IDoc type WMTCID02, message type WMTOCO.

4. After putaway, IDoc WMSUID01 of message type WMSUMO is sent to R/3.

Figure 8 provides a summary listing of these IDoc types.

How are the IDocs sent between SAP and the subsystem? They are sent via a transactional Remote Function Call (tRFC). Every developer who wants to build an IDoc-based interface needs to be aware of the tRFC. Those of you who opt to use an ALE converter, a piece of middleware that does the mapping between the IDoc format and the format in your application, won't need to master the intricacies of a tRFC. The rest of you will, and for this reason I try to unravel its mysteries in the next section.

Mastering the Mysteries of the Transactional Remote Function Call (tRFC)

Again, for those of you who will rely on an ALE converter to perform the mapping between the IDoc format and the format in your application, there is no pressing need to acquaint yourself with the details provided in this section. For the rest of you, my hope is that this information will prove to be very helpful.

To support data exchange between R/3 and the external system, both the MM-MOB and the WM-LSR interface make use of SAP's Application Link Enabling (ALE) technology. You may recall that ALE was introduced in Release 3.0 to enable

Figure 8 Assignment of IDocs to MM-MOB and WM-LSR Interfaces

IDoc	Interface(s)	Designation
WMMBID01	MOB	Goods movements
SDPIOD01	MOB	Transfer of delivery notes to external system
SDPIID01	MOB	Verification of pick quantities for the delivery
SDPAID01	MOB	Verification of shipping unit
WMTOID01	LSR/MOB	Transfer orders (TOs)
WMTCID02	LSR/MOB	Confirming transfer orders
WMCAID01	LSR/MOB	TO cancellation/cancellation request
WMTRID01	LSR/MOB	Transfer requirements (TRs)
WMBIID01	LSR/MOB	Blocking of storage bins (aisles)
WMRRID01	LSR/MOB	Reporting of multiple processing releases
WMSUID01	LSR/MOB	Movement of storage units
WMIVID01	LSR/MOB	Entry of inventory count data
WMINID01	LSR/MOB	Information text

asynchronous coupling of R/3 with external applications such that distributed applications do not have to be tied into a common database. Instead, these applications can rely on messaging to stay in synch with one another. In the case of these two interfaces, synchronization is achieved via the transactional Remote Function Call (tRFC).

To fully understand the workings of the *asynchronous* tRFC, let's step back a moment and review the workings of a *synchronous* Remote Function Call (RFC). An RFC is the call of a function module (routine) into a partner system, where the caller is the RFC client, and the called partner is the RFC server.[1]

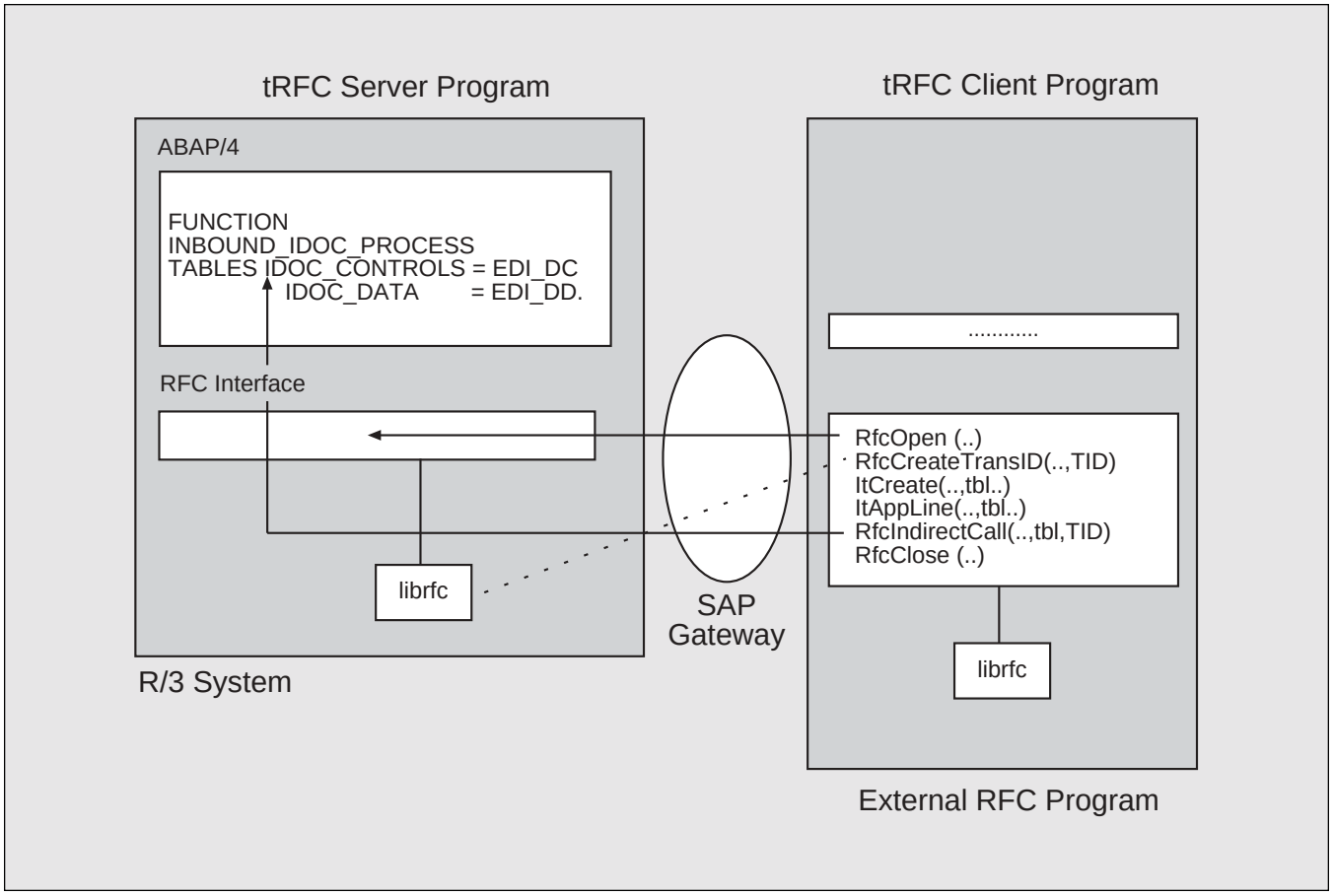
[1] RFC is based on the known RPC (Remote Procedure Call) model from the UNIX-TCP/IP environment. In the SAP environment, RFC is based on a CPI-C interface. The remote call of a function module is easier for an application programmer to use than program-to-program communication, because data is exchanged only using defined parameters. The definition of an application protocol that is necessary, for example, with CPI-C or socket programming, is not required.

Now, let me ask you a question: What happens if a called function module is executed more than once by the server?

Perhaps a communication error occurred while R/3 was sending an IDoc to an external system, and the transaction ID (TID) assigned to this call could not be confirmed, causing R/3 to re-send the data. The external system would first check for the TID and its status, and then decide what to do: accept the data or reject it. This situation can happen if a connection error occurs during a *synchronous* RFC, where the client repeats the call because it has no way of knowing if processing has already occurred. Herein lies the fundamental difference between an RFC and a tRFC. The tRFC is designed to avoid this type of situation and to ensure that the called function module is executed once, and only once.

A simple example will help illustrate the importance of a function module being executed only once…

Figure 9 *Call of a tRFC Client*



Imagine that you report a goods receipt for a purchase order. Let's assume that the ordered quantity was 10,000 pieces, that you have received 200 pieces, but have posted this goods receipt twice instead of just once, so that your inventory information inaccurately reflects a count of 400 rather than 200. Naturally, someone — let's call him John Smith — needs 300 pieces of that material for a delivery. Presuming that a quantity of 400 pieces is available, John arrives on the scene, ready to pick up his 300 pieces. He finds to his dismay that 300 pieces are not available.

Locating and correcting this error can cost a lot of time and money! It's better to leverage the tRFC, and proactively make sure that your application cannot embroil your organization in this type of activity.

The transactional RFC is a *queued* RFC, which means that tRFC calls can be transmitted even if the target system is not available. The data — in the form of an IDoc — simply gets transferred at a later time. A tRFC call is put in a queue on the sending side. It stays in the queue until it has been successfully communicated to the receiving system.

In R/3, automatic job scheduling is used for this purpose. The job starts the relevant function module(s) in the partner system at a later point in time. The data is stored in the tables ARFCSSTATE

Figure 10 The IN BACKGROUND TASK Call Sequence

```
CALL FUNCTION 'INBOUND_IDOC_PROCESS' IN BACKGROUND TASK
DESTINATION 'TEST'
EXPORTING ...
     TABLES IDOC_CONTROL=...IDOC_DATA=...

COMMIT WORK.
```

and ARFCSDATA, and you can administer the data using transaction SM58. (For an external tRFC client, the tRFC client itself must repeat the calls at a later time.)

Calling a tRFC

Take a look at **Figure 9**, which depicts a call from a tRFC client program. You can see that a tRFC client opens a connection to the tRFC server program in the same way that a synchronous RFC client opens a connection — via **RfcOpenEx/RfcOpen**.

The next line of code ensures that the TID is unique worldwide. You use the **RfcCreateTransID** call to read the TID from the tRFC server, which in this example is the R/3 system. (Behind the scenes, R/3 performs an elaborate calculation based on things such as date, time, and installation number to create the TID.)

The actual call of the function module occurs in connection with the API function **RfcIndirectCall** (or RfcIndirectCallEx with RFC library 4.5A or later).

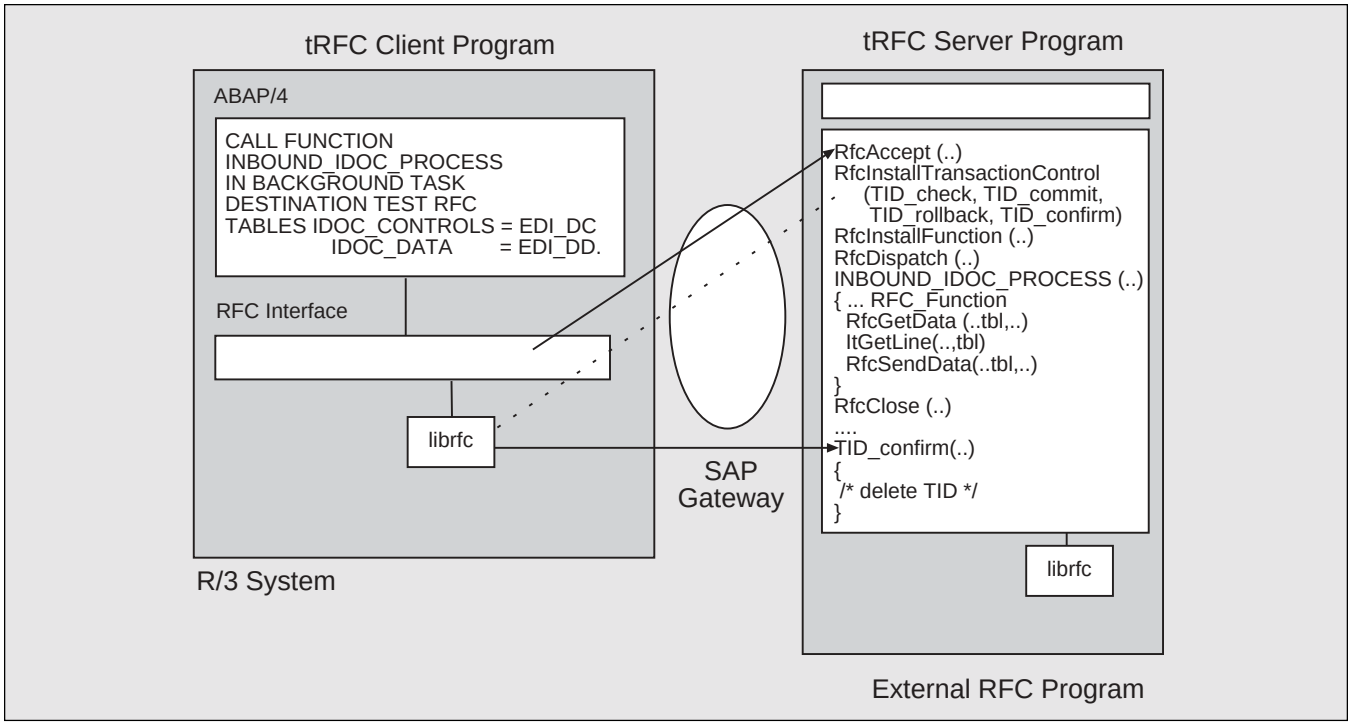
Note that if a connection error occurs during the RfcCreateTransID call, you need to repeat. Otherwise, you would not have a valid TID for communication. If an error occurs during RfcIndirectCall, the tRFC client program must repeat this call with the same TID at a later time. The tRFC client must link this TID with the corresponding data and must not repeat the RfcCreateTransID call. If it did, the one-time transaction execution would not be guaranteed in the tRFC server. After successful execution of RfcIndirectCall, the tRFC client may close the connection.

Unlike the synchronous RFC, the additional call sequence IN BACKGROUND TASK (shown in **Figure 10**) for the CALL FUNCTION statement is necessary for tRFC in ABAP.

The function module that is used for IDoc interfaces like MM-MOB and WM-LSR varies according to the R/3 release. In R/3 3.0 and 3.1, it is the function module INBOUND_IDOC_PROCESS; as of Release 4.0, it is the function module IDOC_INBOUND_ASYNCHRONOUS. (Certain functional enhancements have been implemented for the IDoc Interface in Release 4.0. The length of the names of many of the development objects, for example, has been extended.)

The function module that is used for IDoc interfaces like MM-MOB and WM-LSR varies according to the R/3 release. In R/3 3.0 and 3.1, it is the function module INBOUND_IDOC_PROCESS; as of Release 4.0, it is the function module IDOC_INBOUND_ASYNCHRONOUS.

Figure 11 *Call of a tRFC Server*



Now take a look at **Figure 11**. Here you see that in the tRFC server program, a connection to R/3 is made (**RfcAccept**), the transaction and processing functions are installed (**RfcInstallTransactionControl** and **RfcInstallFunction**), and the dispatch loop is entered (**RfcDispatch**).

The RFC library now calls the function onCheckTid internally. The TID is checked to determine if it has been used before. If it has been used, a message is sent to the R/3 system. If this TID has not yet been used for processing, the corresponding function is called next.

In the tRFC server program, a connection to R/3 is made (RfcAccept), the transaction and processing functions are installed (RfcInstallTransactionControl and RfcInstallFunction), and the dispatch loop is entered (RfcDispatch).

If the processing function is successfully completed, onCommit is executed next, and the changes are written to permanent storage. If an error occurs, the function onRollback is executed, and the changes are rolled back. A positive or negative message is sent to the R/3 system. The R/3 system confirms that the transaction is processed, and the TID administration is updated.

In order to do all this in an external tRFC server program, you must install the following four functions in addition to the actual server functions:

✔ **onCheckTid:** This function is called by the RFC library to transmit the TID from R/3. The delivered TID must be written to permanent storage. Ideally, it is written to a database, but another possibility is storing it in a file. If this TID has never been used before, the return code must be 0. If the TID is recognized as having already been used, the function returns the value <>0.

✔ **onCommit:** If the server function is successfully executed, this function is used for committing changes to permanent storage. Ideally, this occurs using a DB Commit when using a local database. The result is returned to the tRFC client.

✔ **onRollback:** If the server function was not successfully executed, or if there is an error in the RFC library, this function is called to roll back all local changes instead of onCommit. Ideally, this occurs using DB Rollback when using a local database. The result is returned to the tRFC client.

✔ **onConfirmTid:** If a confirmation of the tRFC client is returned, you have to update the local TID administration.

When you call the INBOUND_IDOC_PROCESS function module or IDOC_INBOUND_ASYNCHRONOUS in ABAP, the data in the database is stored. No transmission of data has yet occurred. The data transmission only occurs when a COMMIT WORK is issued. The tRFC server is started, or its registration at the SAP gateway is used.

Processing is subdivided into two phases from the R/3 point of view:

- Phase 1: Function transmission (onCheckTid, call processing function, onCommit)
- Phase 2: Confirmation (onConfirmTid)

If you use tRFC with external programs, you need to know how the TID administration and data processing should be implemented. Files and database systems are available for the necessary permanent storage. Databases enable local transactions to be handled as within R/3 when using tRFC. A tRFC server should ideally implement the TID and data administration using database functions.

Check, commit, rollback, confirm, and processing logic of the tRFC server can be safely performed using the transaction concept of a database system on the external side as well. For a tRFC client, there is a problem in trying to re-send a failed tRFC call at a later point in time. In the meantime, the data together with the received TID have to be stored.

Another problem for TID administration in tRFC clients occurs when several instances of a tRFC client run at the same time. You must solve the problem, whether or not a transaction is currently being processed in another instance of the tRFC client. For a TID with the status CREATED, for example, you need to determine if the program processing this call is still active, or has terminated. If the current tRFC client determines a program failure, it has to re-send the call with the TID of the failed program.

Processing is subdivided into two phases from the R/3 point of view. Phase 1 is the function transmission (onCheckTid,call processing function, onCommit). Phase 2 is the confirmation (onConfirmTid).

Helpful Hints

✔ The most common mistakes made by developers when working with a tRFC for the first time are related to the TID management that needs to be implemented. It happens that an external tRFC client requests a TID at the beginning of each communication process without checking for a pending one. When working with a database on the external side, make sure that you do a local commit within the TIDCommit function.

✔ To be able to communicate via RFC, you need an RFC library for the respective platform. This library is part of the RFC SDK (Software Developer's Kit). The RFC SDK contains executables, libraries, and C programs. Text files explain the SDK setup and give you information on how the programs on

the various platforms are compiled and linked. You must always include the file *saprfc.h* in an RFC program. This file contains the necessary definitions for the RFC API, and is used for handling internal tables. The file *sapitab.h* is also required in our case.

Another component that is needed is the SAP gateway, sometimes referred to as CPI-C handler. You can use either the standard gateway that is installed on every application server, or a so-called standalone gateway on your workstation.

✔ There are several ways to minimize RFC client bandwidth consumption. One way is to open a connection, send the actual data, and close the connection again immediately. Another way is to open an RFC connection, keep the connection open, and close it when you shut down your entire application. Imagine the external system is located in Germany and has a remote connection to an R/3 system in Brazil. If you keep the connection open all the time, your phone company will be very happy.

✔ For problems with RFC server programs, ask yourself the following questions:

1. Are my SM59 settings in order?
   - The used destination must be entered in SM59 with category T.
   - A complete path name must be entered, if possible, for the program to be started.
   - When working with register mode, the program ID must be the same as in your *saprfc.ini* file (case-sensitive!).
   - Make sure that you saved the destination.
   - If you encounter problems, activate the trace. A trace file will be written to the directory where your RFC server is running.
2. Does the PC have an entry in the HOSTS file? On the 16-bit Windows 3.1 and Windows 3.11 platforms, calls to the RFC API functions will produce an "Abnormal Program Termination" if the PC's IP address and hostname are not in the HOSTS file. Exceptions are PCs running the Microsoft TCP/IP protocol stack, or PCs running Windows 95 and Windows NT using the native TCP/IP protocol stack.
3. Does the PC have a valid SERVICES file? Does it contain sapdp00-sapdp99, sapgw00-sapgw 99, sapsp00-sapsp99, and sapms<SID> of the customer's R/3 system ID(s)?

✔ Useful transactions for troubleshooting include:

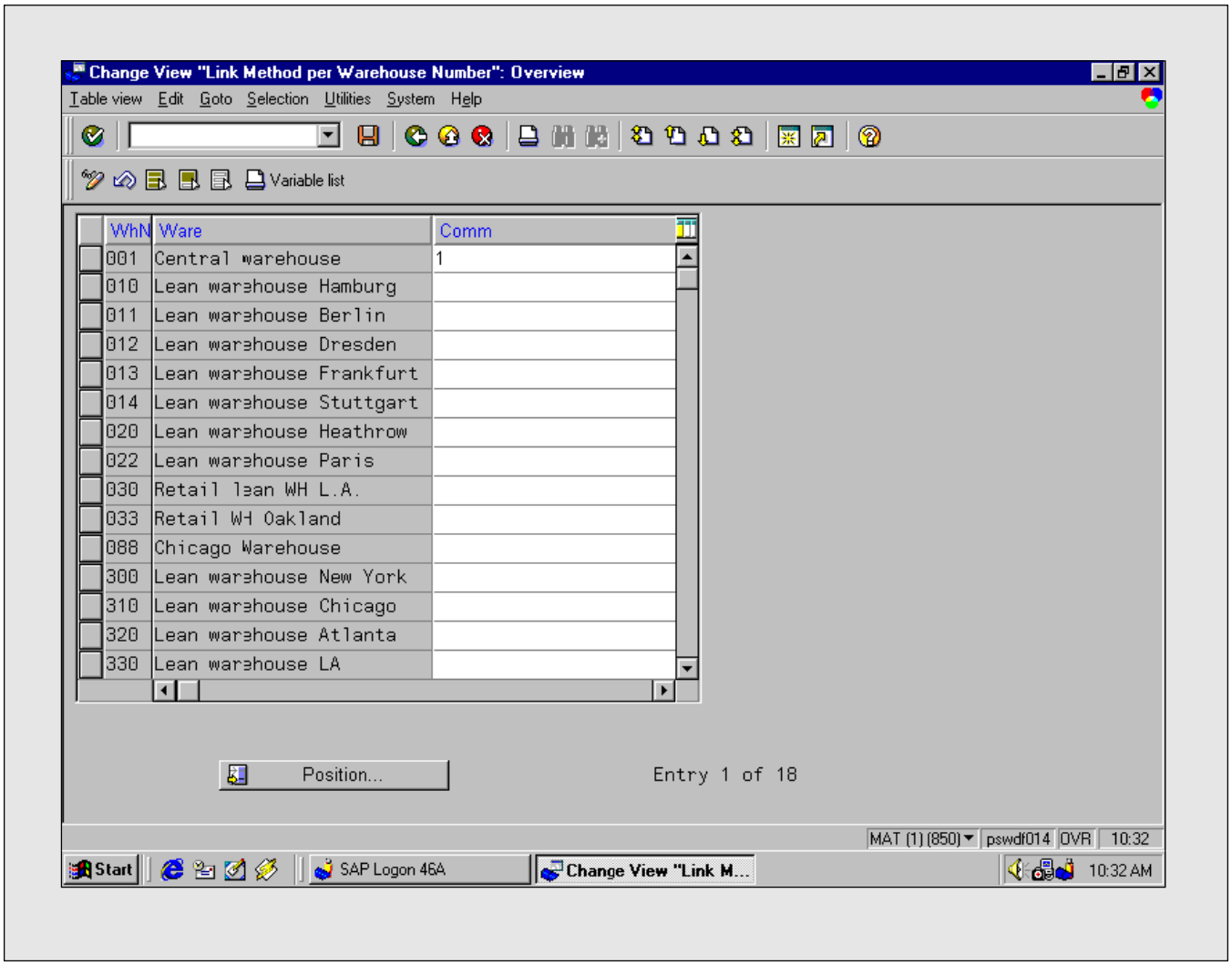
- SM21 — system log
- SM58 — monitor for tRFC
- WE05 — IDoc monitor
- SMGW — gateway monitor; check the status of your server, display gateway trace

✔ If you receive an error (e.g., message XY123), you can look up its meaning in R/3 in table T100.

Customizing the WM-LSR and MM-MOB Interfaces

Most of the customizing activities performed for the WM-LSR and MM-MOB interfaces are the same. So I will focus on just one here, the WM-LSR interface. The only thing you need to remember when customizing the MM-MOB is that with this interface, unlike the WM-LSR interface, you have to maintain the output determination in the SD module for the transmission of deliveries.

Figure 12 Specifying the Communication Method



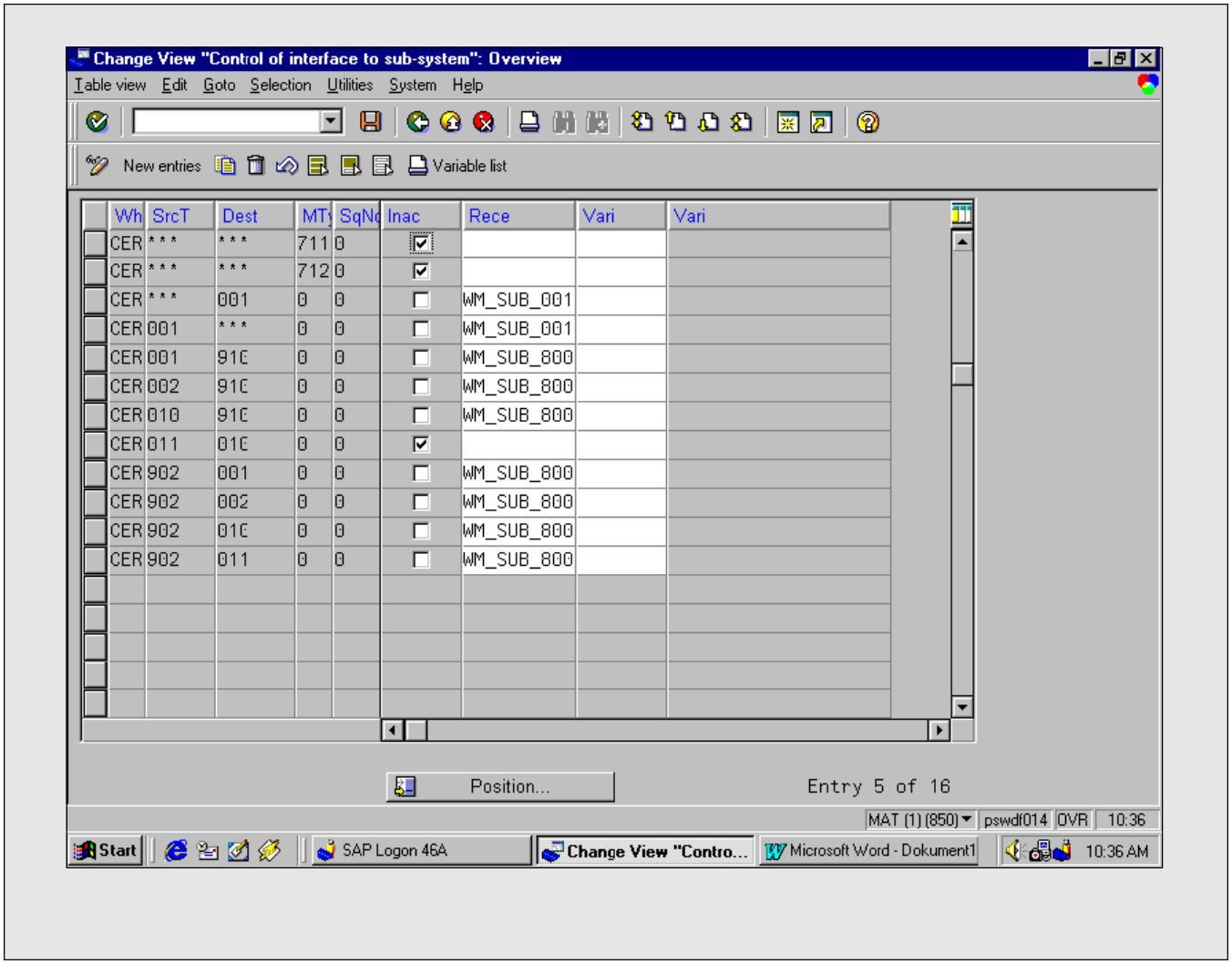
Almost all customizing settings for the WM-LSR interface can be made from transaction OMKY, as follows:

1. First, you activate the interface for your warehouse number(s), and specify the communication method (in our case "1-Communication via ALE"), as shown in **Figure 12**. Note that the definition of message types and variants are optional.

2. Then you assign the function modules for IDoc creation — i.e., you specify the name of the function module that should be executed for a message type. For example, you link message type WMTORD to function module L_IDOC_CREATE_WMTOID01.

The only thing you need to remember when customizing the MM-MOB is that with this interface, unlike the WM-LSR interface, you have to maintain the output determination in the SD module for the transmission of deliveries.

Figure 13 Establishing the Definition of the Interface Control



3. Next, and this is very important, you establish the definition of the interface control, as shown in **Figure 13**.

4. Within the interface control, you specify all combinations of warehouse number, source storage type, destination storage type and movement type, whether an IDoc should be created, and where that IDoc should be sent to, in this way:

```
ABC   902   001   501   0   EXT_SYS
```

Here, in warehouse number "ABC," whenever a transfer order is created that goes from storage type 902 (the goods receipt area) to storage type 001 (high rack storage) for movement type "501," IDoc WMTOID01 is created and will be sent to the logical system EXT_SYS.

If you want to send data to multiple external systems, you need to create a line for each system and increase the field sequence number, which, by default, is set to zero.

At this point, you are ready to do the ALE customizing from transaction OMKY, as follows:

1. **Define a partner number (logical system).**

2. **Define an ALE port for the type tRFC.** In Release 4.0 and later, you can specify the type of control records that should be created. The value "2" stands for control records of R/3 Release 3.0/3.1, and results in the call of the function module INBOUND_IDOC_PROCESS, whereas "3" means control records for R/3 4.x, and will lead to the execution of the function module IDOC_INBOUND_ASYNCHRONOUS.

3. **Define an RFC destination.** Within your RFC destination, you tell the system where your application can be found and when it should be started. You create a destination of type "T" (communication via TCP/IP) and select an activation type. "Start" means that R/3 will start your program, send the actual data, and terminate your program again. In this case, you need to specify where, exactly, your executable files are located: on an application server, or on your frontend workstation. An alternative would be the activation type "Registration," with which it is possible to register your application at an SAP gateway and wait for data in a dispatch loop. So all the communication overhead for setting up an RFC connection and closing it after the data has been sent is no longer there, which can speed the processing up to a factor of 10 — so we really recommend that you work with Registration.

4. **Configure a partner profile.** Within the partner profile, you specify the IDocs that are exchanged for your logical system. The terminology is from the point of view of R/3: outbound parameters refer to message types that are sent from R/3 to the external system, and inbound parameters are sent from the external system to R/3. For outbound parameters, it is necessary to specify the ALE port that should be used, and the output mode (whether IDocs should be transferred immediately, or whether they should be collected and sent in packages).

5. **Lastly, the ALE distribution model must be maintained.** The message types for the direction R/3 → external system are:

WMTORD	Transfer orders
WMCATO	Cancellation request for transfer orders
WMRREF	Release reference numbers
WMINVE	Inventory bin location list

ALE looks for the outbound partner profile to determine communication parameters. The outbound partner profile indicates the port to be used. The port is connected to an RFC destination that is used to connect to the receiver (in our case the external system).

The message types for the direction external system → R/3 are:

WMTORD	Transfer orders
WMTOCO	Confirmation of transfer orders
WMCATO	Cancellation of transfer orders
WMSUMO	Move storage unit
WMINVE	Inventory count data
WMBBIN	Blocking/unblocking of storage bins
WMINFO	General information text
WMTREQ	Transfer requirements

Helpful Hints

✔ When working with third-party products and systems (e.g., bar-code devices, radio-frequency devices, fork-lift control systems, picking systems, etc.), be sure to use products that have been certified by SAP. As of today, there are about 42 interfaces for which certification is available. A list of these interfaces, as well as a list of certified partner

products, can be found on SAP's Web pages at **http://www.sap.com/csp**.

✔ The most common mistake made by developers when setting up IDocs is the initialization with hex0 or hex1. All the fields need to be initialized with blanks.

✔ In the partner profile you define as one step of the ALE customizing, use output mode "Collect IDocs" and specify a reasonable package size (e.g., 50-100) for the outbound parameters. Otherwise, the external application will have to process 50 IDocs, one after the other, with 50 TIDs to update and manage, instead of a package of 50 IDocs with the same TID.

Conclusion

In future releases, look for three improvements to the MM-MOB/WM-LSR application development landscape we just discussed:

- Delivery handling will change. There is a new generic IDoc type, DELVRY01, which can be used instead of SDPICK (SDPIID01) and SDPACK (SDPAID01).
- The LES (Logistics Execution System), which was introduced in Release 4.5, combines the functionality of the warehouse management, shipping, and transportation, and will have an impact on the interfaces in the logistics environment.
- With Release 4.6, there is direct radio frequency support in the warehouse management. RF terminals receive data directly from the R/3 system, and can send data back the same way.

And for continual updates about these issues, be sure to check the R/3 release notes and SAP's Development News on our Web pages.

Michael Ottenstein studied "Wirtschaftsinformatik" — a combination of business administration and computer science — at the University of Mannheim, Germany. He joined SAP AG in 1996 as a member of the Integration & Certification Center, where he gained experience with SAP's communication technologies. Since 1997, he has been responsible for the certification of SAP's interfaces in the areas of MM, WM, and SD. Michael can be reached at michael.ottenstein@sap.com.

