
1 Introduction

Over the last few years, data warehousing has developed into one of the central topics in IT. It is used as a strategic tool to provide information for controlling and decision-making processes. Further, data warehousing provides the basis for other technologies—e.g., in the areas of strategic enterprise management, customer management, and process analysis.

The need for systems to provide and analyze information is as old as information technology itself. Initially, data analysis was understood to be an appendix to operating systems. In the mid-90s, the term *data warehouse* won recognition and turned into a separate area of IT with its own specific concepts since data warehousing systems not only enable the analysis of large volumes of company-wide data but also reduce the complexity of data provision.

Data warehousing systems centrally provide data for controlling and decision-making processes. Technical and functional characteristics make each and every data warehousing system unique.

Data warehousing systems may be classified in two categories:

- Operational administrative systems
- Decision support systems

Operational Administrative Systems

Operational administrative systems (also called *transactional systems*) provide functionality for the enterprise to administer and execute business transactions. The typical tasks of an operational administrative system are order entry, invoicing, inventory management, personnel administration, and payroll. The so-called enterprise resource planning (ERP) systems support not only individual areas but all functions of the enterprise value chain. An example of an ERP system is SAP ERP Central Component (SAP ECC) or its forerunner, SAP R/3. In the following, they are both referred to as SAP ERP.

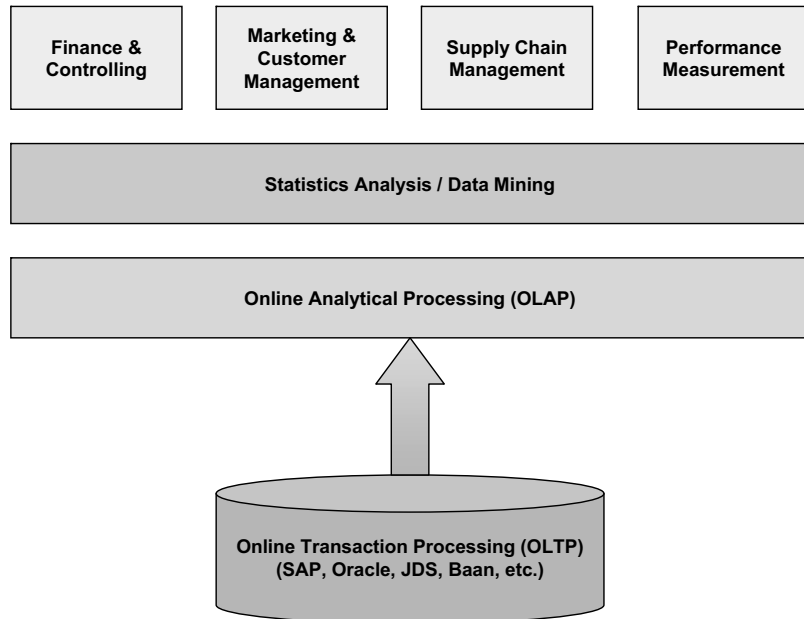
Since operational administrative systems technically distinguish themselves by showing functions based on individual transactions (orders, booking entries, etc.), they are also called online transaction processing (OLTP) systems.

Decision Support Systems

Given their individual transaction focus, administrative operational systems do not form a suitable basis for complex business decisions. Hence, there is a counterpart—*decision support systems (DSSs)*.

A DSS systems are classic reporting and interactive data analysis systems (*online analytical processing, or OLAP*) as well as systems used to search for complex coherences and unknown data patterns (data mining).

Figure 1-1
Decision support systems



Apart from the tools providing general data analysis functions, there are process-oriented tools specializing in the analysis of particular process data. They are typically used in finance & controlling, marketing & customer management, supply chain management, and performance management (see figure 1-1).

Data Warehouse Systems

Any decision support system requires a data pool on which analytical functions are based. OLTP systems do not form a suitable data pool given their functions and especially their data structures. Usually, DSSs do not have their own data management but may share a data warehouse system with other DSSs.

DSSs are mainly characterized by their data management, which strongly supports the analytical review of the data. The content of a data warehouse is extracted from the OLTP system data and transferred to data models in BW, which are optimized to analyze large databases. It is kept redundantly from source system data.

So, a data warehouse is not a complete decision support system per se but only the basis for such, and it offers the respective interfaces. However, software producers usually couple data warehouse systems at least with an OLAP tool. From this habit, data warehouse systems are often incorrectly called OLAP systems since they only describe sales habits of suppliers.

With SAP Business Information Warehouse (SAP BW), the Business Explorer tool links a complete product suite to decision support systems and they can be part of a BW implementation (e.g., the Data Mining Workbench and large parts of Integrated Planning).

Content

As the title says, this book deals with SAP BW as a data warehouse and as such ignores the BW-integrated decision support tools as far as possible. This is not because these tools may be irrelevant, but an adequate description of such tools would fill a separate book and is beyond the scope of this one.

The book is divided in six sections:

- Architecture
- Data Model
- Analytical Engine
- Extraction and Staging
- BW Design
- BW in Live Operation (also referred to as Productive Operation)

Within the section on architecture, BW installation components and their functions and interrelations are discussed conceptually. The Metadata

Repository, the application platforms of the base system as well as their communication technologies are outlined.

Then, in the next section, the focus is on specific data models in data warehousing in general in order to deduce and explain the specific implementation of the *data model* in SAP BW. We'll discuss performance-relevant aspects of data modeling and give you practical ground rules for modeling.

Between the decision support systems and the data in BW operates the analytical engine. It receives queries, reads the BW data, and considers particular BW data modeling as well as status information from extraction and staging. The respective section describes how the analytical engine works and how it is tuned or monitored.

The section on extraction and staging details how the data flow from source systems to BW can be defined. The description follows a reference architecture that partitions data flow in logical levels in which they are validated, transformed, error-corrected, and integrated. Further, data quality and performance tuning are considered from a staging angle.

The next section is on BW design. In this section we'll discuss typical modifications of the reference architecture, which was discussed in earlier sections. Here, the focus is on partitioning techniques and large-scale architectures.

The section on live operation gives an overview of all regular processes to be executed. And considering organizational as well as technical aspects, I'll explain how automation and monitoring are realized.

The book ends with a comprehensive appendix, dealing with special topics such as currency conversion, transportation, and migration as well as development of metadata content.

I Architecture

A data warehouse architecture usually describes data-warehouse-related components, their functions, and how they cooperate on a conceptually comprehensive level. However, a data warehouse cannot be considered isolated but it is integrated in a system landscape with components other than SAP BW. Chapter 2 describes SAP BW *architecture components* based on a three-layer model; the discussion will center on the particularities of BW architecture, but the components of connected systems will also be considered.

The basic understanding of the underlying application platforms and their communication interfaces is key to building data warehouse architecture with SAP BW, as outlined in chapter 3 on the BW basis system.

Special focus lies not only on data storage but also on data administration and data structures in BW, as described in chapter 4, which is about the Metadata Repository.

2 Architecture Components

As a data warehouse, BW is always used in combination with other system components, which fulfill specific functions in extraction and decision support. When all functional areas of BW as well as the connected systems are combined, BW shows a three-layer architecture:

- Extraction layer
- Data warehouse
- Decision support systems

These three layers are outlined in a separate figure in the fold-out together with their components. They will be described and explained in the following sections.

2.1 Extraction Layer

The extraction layer describes the different functional systems from which SAP BW extracts data as well as the respective extraction procedures.

The extraction of analysis-relevant data from source systems is a central issue of each data warehousing project. Complex data structures, large volumes of data, and volatility of source system data (e.g., subsequent change of supplied data) may cause extraction to become a very extensive part of BW that you should not underestimate.



With its communication interfaces (see section 3.1), BW uses several powerful tools to access source system data where the type of source system will specify the interface to be used.

Thanks to its own communication interfaces, BW can access the following source systems:

- SAP ERP
- SAP BW systems
- File systems
- Database systems
- XML documents (in SOAP messages)
- Source systems with the Java Database Connectivity (JDBC), OLE DB for OLAP (ODBO), or XML for Analysis (XML/A) interface

Should the BW extraction options not meet requirements, you can use extraction tools from other suppliers (referred to as *third-party ETL tools*).¹

For third-party ETL tools, BW offers an interface in which it considers the third-party ETL tool to be the source system. Depending on the source system type, these tools use a large variety of extraction methods in alignment with the source system. Note, however, that the BW extraction options have been considerably enhanced in the most recent releases and the use of third-party ETL tools has become almost obsolete.

In the following sections, we'll explain how to access the source system types. You'll find detailed descriptions on how to create an extraction from the source systems or in BW in chapter 14 and 15.

SAP ERP systems

In practice, SAP ERP systems are the most relevant type of source system for the BW extraction layer, which explains the extensive extraction options that SAP offers for this source system. The extraction is effected by so-called extractors, which can be installed as plug-ins from version 3.0D of the SAP ERP systems onward.

Extractors offer not only the necessary interfaces and extraction programs to technically enable extraction, but also preconfigured extraction scenarios for the very different modules (called BI content, as described in appendix D).

Preconfigured extraction scenarios can be customized and individual extraction mechanisms can be developed. Technical background information on extraction can be found in section 14.2.

BW systems

When you're building large-scale architectures (see chapter 27), the extraction of data from BW systems plays a special role. From a technical point of view, extraction from BW systems is very similar to extraction from SAP ERP systems; in BW, there are also extractors that meet the technical requirements to extract data from BW and to pass it on to other BW systems.

1. A current List of certified third-party suppliers can be found in the SAP Service Marketplace.

The communication between BW and a source system always requires a joint communication interface. A limiting factor is the organizational and technical authority over control of the interface, which has to be accepted.

File system

The lowest common denominator of data exchange between heterogeneous systems often is making data available in ASCII files on the file server. There are various disadvantages to this, especially in terms of exchange of metadata and common control information. However, it is often the easiest way for the source system to provide data in a file format.

Since its first release, BW uses a file interface that is well suited for exchanging large volumes of data. The only requirement for a “BW-suitable” file is a flat file structure design; that is, each record in the file needs to show the same composition. Reading of hierarchical data structures is not possible within the standard options of BW.

For several special database systems, BW offers direct access to tables and views on databases with an interface called ***DB connect***. Access is not accomplished via a JDBC or comparable interface. Instead, BW functions as a database client to access the database systems, which results in high-performance access.

Database systems

Usage of DB connect requires certain libraries and clients that exist only for database systems that BW can use to store its own data. However, even within this selection, a database system cannot be extracted in every given constellation; especially if BW runs on a UNIX platform, libraries and clients are not available to the various database systems.

Before planning to use DB connect, you should verify that it can be used with the combination of the system platform underlying BW and a database system to be extracted.

To support the so-called “open standards”, BW offers a web service. This is the SOAP service, which allows you to supply BW with XML documents. However, the process of doing so is so specific that the SOAP service may hardly be called an open interface.

XML

The protocol of the SOAP service is rather tailored to the Exchange Infrastructure (SAP XI) technology, which has been created for data exchange between heterogeneous systems. It only makes sense to use XML documents for extraction when an XI system is operating between the original source system and the SOAP service. In fact, it is possible to migrate XML data to BW without an XI system. However, the effort to comply with the protocol requirements is so high that the use of the interface is questionable.

In any case, one has to consider that the SOAP service is the slowest of all interfaces (by far!), and ideally, it should be used only to exchange individual records.

JDBC, ODBO, XML/A

Describing the conglomerate of the very different interfaces JDBC, ODBO, and XML/A in one step might seem slightly odd. This grouping results from the change in architecture SAP has realized with the new application platform, the Java™ 2 Platform Enterprise Edition (J2EE) server, whose connection framework allows access to these interfaces via the ***Universal Data Connect (UDC)***.²

JDBC, ODBO, and XML/A provide access to the J2EE server's connection framework rather than access to the source systems. This is a component of the J2EE engine configuration.

Actually, the J2EE server is a full-fledged application server. Apart from the named interfaces, it may also dispose of components such as Portal, Content Management, and TREX, as well as its own administration environment, the J2EE Visual Administrator. The J2EE server will be addressed in chapter 3, with the description of the BW basis system.

From a BW point of view, the J2EE server is only relevant due to its UDI Java components, and thus—here and in the following—it will be considered only as part of the extraction layer.

2.2 Data Warehouse

The data warehouse layer is the heart of BW and basically covers the Data Manager and the staging engine.

Data Manager

The scope of the Data Manager comprises the administration of specific data structures of SAP BW, the management of data content, and the provision of access to a database.

The *definition of data structures* is achieved in BW on an abstract level in the form of object definitions for master and transactional data (e.g., InfoObjects and InfoCubes). The data structures of the database system where the data will be stored are derived from these object definitions.

2. The terminology differs. The terms *Universal Data Connect (UDC)* and *Universal Data Interchange (UDI)* are used. In this book, I'll use the more common term *Universal Data connect*, or *UD connect*, if UDI does not characterize a special term (e.g., UDI Java components).

Thus, the Data Manager acts as an intermediary between the definition of BW objects in the metadata and the technical storage of data in the related database systems.

The *management of data content* deals with physical access to the databases, and in BW it is the intermediary between the diverse BW functions and the database access. In particular, the Data Manager has to be able to handle the specific data structures of BW.

Database-specific definitions and operations are also included in the scope of the Data Manager; e.g., creation/deletion of indexes, configuration of partitioning, handling of Stored Procedures, and so on.

Thus, *providing external access* is also in the scope of the Data Manager. If BW is used internally to access databases, it is essential that the Data Manager enables analysis tools to access to the BW system.

Access is made using a central interface, the so-called analytical engine, whose task it is to transform and optimize incoming queries in a BW-internal format and also to return result sets. However, the process and quality status of the databases, as well as possibly limited authorizations, have to be considered.

Before the Data Manager can handle any data, the data must have been received and prepared by the extractors of the respective source systems. The preparation is necessary as the source systems usually do not follow any identical conventions regarding data storage (e.g., different use of upper and lower case, handling of leading zeros, different keys to same data) or the data may not have been delivered in the way the BW is supposed to provide it to the user (e.g., calculation of key figures).

Staging engine

Extraction from the source systems and the subsequent controlling and monitoring of data streams is initiated through staging. To accomplish the extraction and processing, the staging uses data structures that are usually temporary but may be filed persistently for quality assurance purposes.

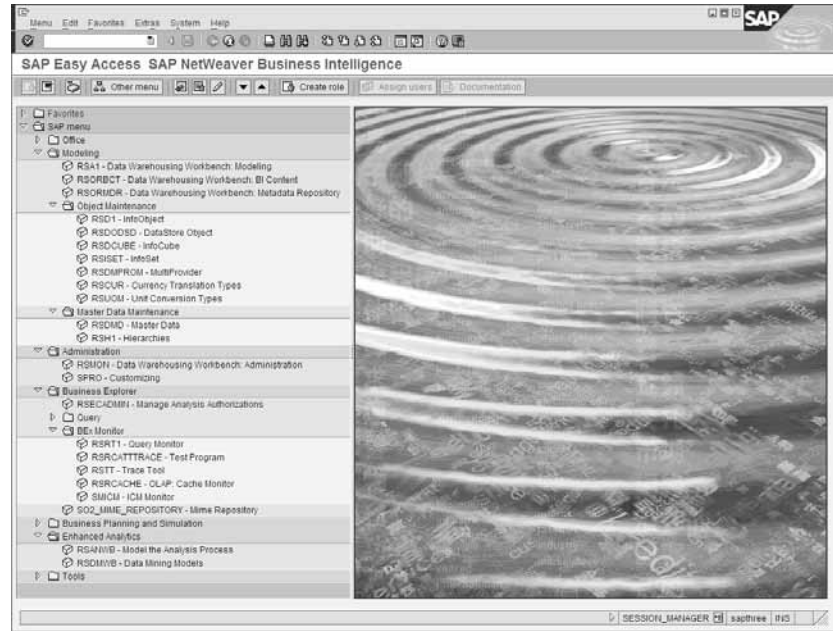
All settings regarding definition, controlling, and monitoring of the Data Manager and the staging engine are filed in a central Metadata Repository, whose structures will be explained in detail in chapter 4.

Data Warehouse administration

In a way, the Easy Access menu serves in SAP BW as a user interface to the Metadata Repository, and it opens right after the SAP graphical user interface (SAPGUI) logon³ (see figure 2-1).

3. The Easy Access menu is an area menu RS00_BW, which can be activated systemwide by using the transaction SSM2.

Figure 2-1
SAP Easy Access menu



© SAP AG

However, it is wrong to consider the Easy Access menu to be an interface to the Metadata Repository since the Easy Access menu is mainly a structured compilation of menu items that branch out to further BW transactions. There are numerous transactions to control and monitor the system, but there are *also* transactions to maintain the metadata objects in BW.

The transactions can be started either through the menu items of the Easy Access menu or through the direct indication of the transaction in the respective entry field, as shown in figure 2-1 (upper left).

Data Warehousing Workbench

The pivotal part of the BW is the *Data Warehousing Workbench (DWWB)*. It can be accessed from the Easy Access menu, or alternatively through the transaction (see figure 2-2).

The majority of metadata for BW objects is defined in the Data Warehousing Workbench. Furthermore, the DWWB also contains functions to control and monitor administrative processes in the Data Manager and the staging engine.

The majority of explanations in this book refer to settings that are made in the Data Warehousing Workbench.

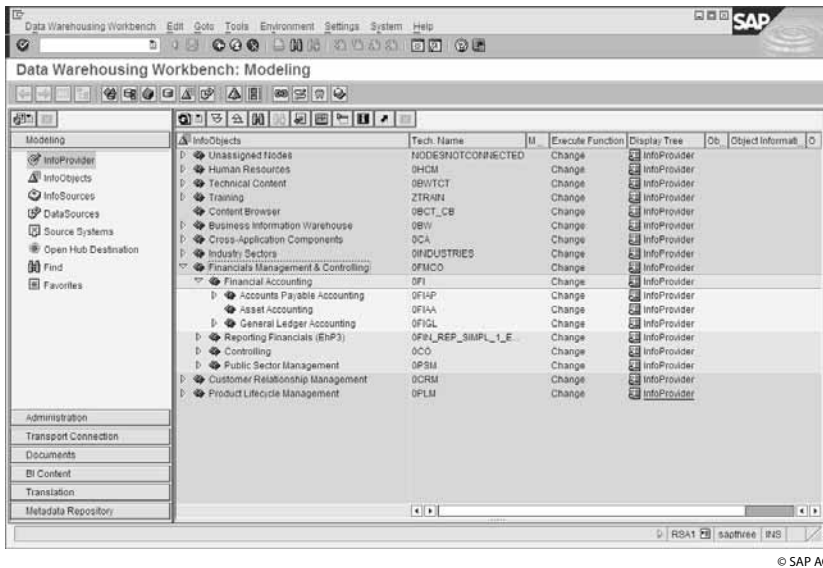


Figure 2–2
Data Warehousing
Workbench

2.3 Decision Support Systems

The Decision Support layer contains all tools with which data of the BW can be retrieved and prepared to be relevant to decisions. The data is accessed using the MDX processor of the BW, which acts as intermediary between the Data Manager and the query logs of the access interfaces.

The following interfaces can access the BW via the DSS tools:

Interfaces

- The SAP-specific *Business Application Programming Interface (BAPI)*. Up to version 2 of BW, it was the only open interface for DSSs and thus it is the most common interface. SAP's own DSS tools also use BAPI, but they can also access the BW without the BAPIs, which no other DSS tool is allowed to do.
- The HTTP-based BEx-Service to provide websites with HTML code and JavaScript for *Web Queries* and *Web Applications*.
- The HTTP-based *XML/A service* to provide multidimensional data using the platform-independent standardized XML protocol XML/A (XML for Analysis).
- The Microsoft-specific *ODBO interface*⁴ to provide multidimensional data based on the COM protocol.

Business Explorer In addition to the option to supply any DSS tool with data, BW uses a range of its own decision support tools that access BW via interfaces from the outside, exactly as third-party tools do. In BW, these tools are pooled in the product suite *Business Explorer (BEx)*. They contain tools for web- and Excel-based OLAP analysis, preparing and sending formatted reports, developing web-based analytical applications, data mining, running planning applications, and so on.

Until release BW 3.x, the Business Explorer tools could mostly be used independently, but in release 7, some of them can be used only in combination with the NetWeaver 2004s portal. This is the case, for example, for broadcasting or when deploying ad hoc analyses on the Web.

4. ODBO stands for OLE DB for OLAP (Object Linking and Embedding Database for Online Analytical Processing)

3 BW Basis System

SAP NetWeaver 2004s, and therefore also other SAP products (such as BW or ERP), is based on a three-layer client/server platform consisting of database layer, application server, and client.

The application server is the *SAP Web Application Server*, which we'll refer to as Web AS. It is a *development and runtime environment* at the same time and has all the components of a typical application server, such as database and communication interfaces, lock management, and job and process control, as well as tools for system administration.

Application server

The execution of processes in SAP BW is based on the runtime environment of Web AS, in which the functionality was developed. It can also be enhanced by proprietary development through its integrated development environment.

For scaling and load balancing, it is possible to use not only one but several application servers to direct dialog and background processes to the application server that has the least load when the process is started.

The use of several application servers for load balancing is an effective tool to scale operational systems like SAP ERP. With systems such as SAP BW, the bottleneck is usually not the application server but the database server; the use of several application servers for performance tuning often (but not always) fails to produce the desired effects.



Up to this point, Web AS is similar to other application servers. However, from version 6.40⁵ onward, Web AS not only had one development and one runtime environment, it had two of each, one for ABAP⁶ (**A**dvanced **B**usiness **A**pplication **P**rogramming, originally **A**llgemeiner **B**erichts-

-
5. The versioning of SAP BW and the respective basis system are different. The Web AS in version 6.40 was first released in SAP BW in its BW 3.5 version.
 6. In this context, ABAP refers to the programming languages ABAP/4 as well as ABAP objects.

Aufbereitungs-Prozessor = general report creation processor) and one for Java programs.

Each of these Web AS components is a self-contained application server, and they can be used technically isolated from each other. Thus, both Web AS components are full application servers that can be run on identical or different physical hardware.

Web AS ABAP

The so-called Web AS ABAP is the actual SAP BW platform. It is an enhanced basis system, comparable to the “old” SAP R/3. So, Web AS ABAP forms the consistent platform for the majority of SAP products.

As in releases prior to 3.5, the data warehouse functions are solely based on WEB AS ABAP; i.e., to run BW 3.5, the Web AS ABAP part needs to be installed, whereas its counterpart, WEB AS Java, is only an optional component of the BW installation.

Web AS Java

From BW 3.5 onward, the Web AS Java is a new component of the BW basis system. It is not an enhancement of Web AS but a completely revised runtime environment or platform, which can be used in coexistence with the existing Web AS ABAP.

The basis for this platform is the J2EE Engine, which is compliant with JMX 1.3 and on which JavaBeans, servlets, JSPs, JNDI, JMS, and Java Mail can be implemented in accordance with the agreed-upon standards.

Database layer

On both platforms, access to data content is encapsulated by a persistence layer that controls all access to the database.

With Web AS ABAP, the persistence layer is the *ABAP Dictionary*, which can be accessed through the SQL dialect Open SQL. Due to its relevance for SAP BW, the ABAP Dictionary will be further explained together with the Metadata Repository in chapter 4.2.

The respective counterpart in Web AS Java is the *Java Database Dictionary*, which can be accessed with Open SQL APIs. However, it is irrelevant for data warehouse application development in SAP BW 3.5.

In both cases, one of the established relational database systems, such as Oracle, MS SQL Server, DB2, Informix, or Max DB (formerly SAP DB or ADABAS D) is used for the actual data management. The Web AS itself stores programs and data structures but not the data content.

Since there is no comprehensive lock mechanism for Web AS ABAP and Web AP Java and individual storing formats are used, the two platforms do not share the same database. Instead, each application server has its own database schema so that the separation of Web AS ABAP and Web AS Java is continued technologically.

Unlike the application layer, the database layer cannot be assigned to several servers,⁷ and with high data volume requirements, the database will most likely be the bottleneck in the BW system. When selecting hardware, it is very important to choose the right database server for Web AS ABAP.

There are two hearts beating in Web AS; that is, two application servers. They may coexist, but strictly speaking they do not have much in common because they are using different programming languages and cannot even share a database. Thus, balancing similar tasks on the two servers is impossible—a gradual migration of functionalities from one part of the Web AS to the other is also rather utopian.

Who does what?

The existing data warehouse functionality of SAP BW—and its enhancements—will inevitably remain in the scope of Web AS ABAP. Web AS Java will take over new tasks that typically focus on the data exchange with other systems; after all, with the J2EE Engine, Web AS is to open up to so-called “standards”. These are products such as SAP Enterprise Portal, SAP Mobile Infrastructure, Knowledge Management, and SAP Exchange Infrastructure (SAP XI). All products that have been developed on a Web AS Java basis have been bundled with SAP BW in a single product portfolio and have launched as the integration platform *NetWeaver*.

From an SAP BW point of view, the J2EE Engine in Web AS has two purposes: On the one hand, it controls the portal, which is mandatory for several development activities around data analysis; e.g., for ad hoc reporting or the Information Broadcaster. On the other hand, it may control the Universal Data connect (UD connect), which is used to extract data from JDBC, ODBO, and XML/A sources to be handed over to BW (see section 2.1).

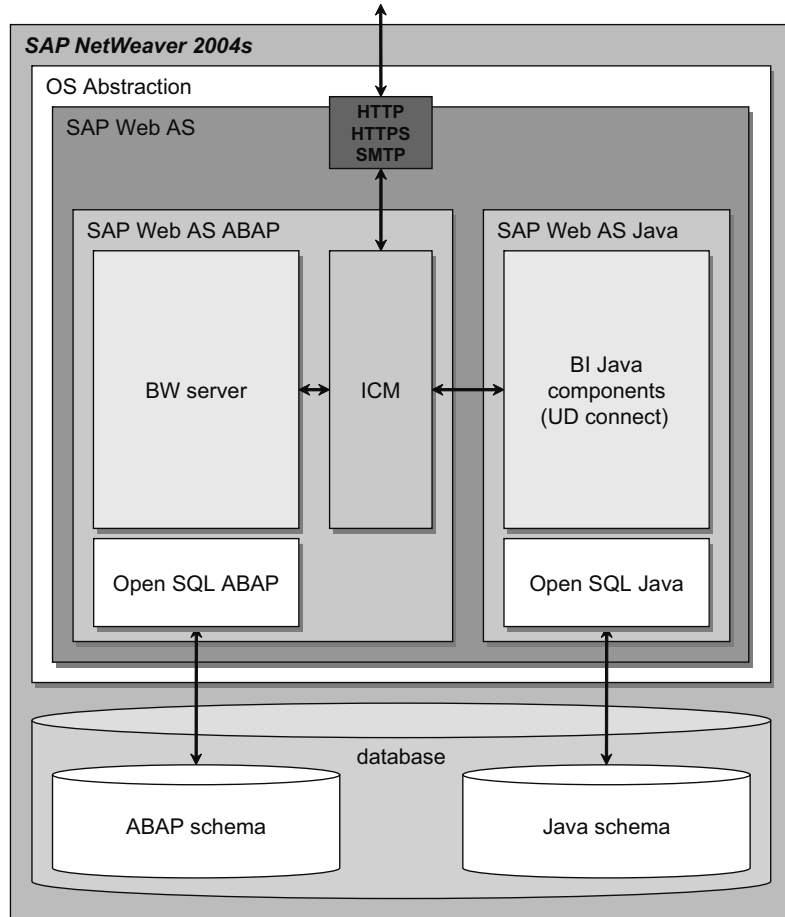
Figure 3–1 shows the architecture of the BW basis system.

The description of Web AS ABAP and Web AS Java has established that these two Web AS parts are technologically self-contained system platforms. However, they cannot be called isolated—quite the contrary!

The communication interfaces, which are very important in SAP BW, will be discussed separately in the following chapter.

7. An exception is the use of the Real Application Cluster (RAC) in Oracle databases. But even then, the database itself will take on the load balancing on several physical servers and not Web AS.

Figure 3-1
BW basis system



3.1 Communication Interfaces

To exchange data with source systems or decision support systems, BW uses a range of communication interfaces. Even though Web AS Java should be used for communication and integration, Web AS ABAP also has communication interfaces. Given the history of this product, the communication interfaces in Web AS ABAP are far more important in practice. Furthermore, Web AS ABAP and Web AS Java not only communicate with other systems but also among themselves.

The basics of the communication technology will be described in this section. A detailed explanation is beyond the scope of this book. For a BW

novice, a discussion of the communication interfaces might not be relevant at this point, and it may be used as reference at a later stage.

The fundamental communication technologies of Web AS are as follows:

- Web AS ABAP's *file interface* for communication with heterogeneous systems as well as the *Business Application Programming Interface (BAPI)* to communicate with SAP-compliant systems
- *Internet Communication Framework* to provide web services in Web AS ABAP
- *J2EE Connector Architecture* to develop Java-based interfaces in Web AS Java

At this point, we are not concerned with the data warehouse functions of BW. We are only providing a basic overview, which can be expanded in the following chapters.

3.1.1 File and BAPI

The file interface and BAPI are both technologies that were used in SAP R/3 to communicate with other systems.

In BW, the file interface is used mostly to exchange data with non-SAP systems—e.g., extraction from source systems or migration of edited analysis data in hub and spoke architectures (see chapter 27).

File interface

In heterogeneous system environments, the file interface often is the lowest common denominator of all systems, and SAP R/3 systems only communicated with each other via file systems prior to release 3.

For systems that are SAP compliant in their communication with SAP BW and SAP ERP systems, SAP designed the Business Application Programming Interface (BAPI). It is a series of interfaces that can be used by SAP itself as well as by third-party suppliers to access SAP systems. The interfaces are defined and open at an application level so that they will remain stable in case of a new BW release by SAP.

BAPI

Technically, the process of starting a BAPI is a so-called Remote Function Call (RFC). The Remote Function Call is one option you can use to start a function module in a different SAP Web AS ABAP. This can be done from an SAP ERP or BW system or from proprietary programs. The data is transferred via TCP/IP or X.400 as byte flow.

Remote Function Call (RFC)



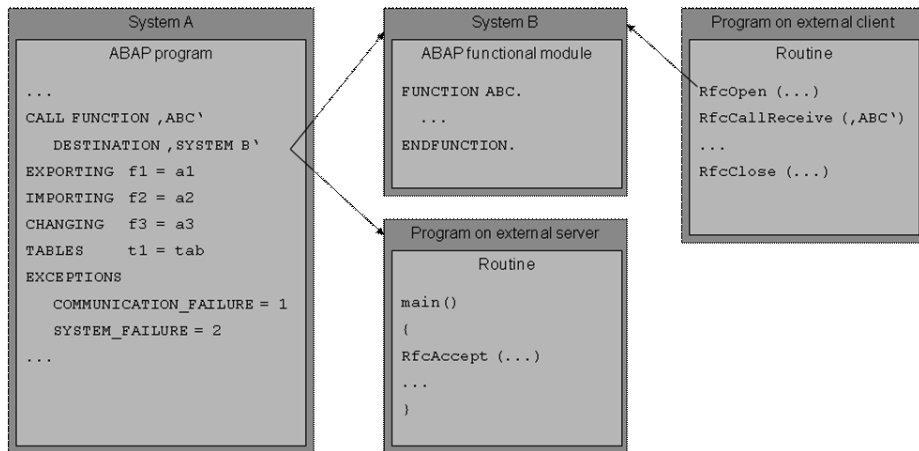
Even though starting a BAPI is technically considered a Remote Function Call, the Remote Function Call does not need the open function modules to exchange data. Via RFC, undocumented and nondisclosed function modules can also be started, in the same way as via BW extractors or the DSS tools of BEx. So systems that rely on BAPIs (for certification or otherwise) always have a disadvantage compared to proprietary SAP tools that can bypass BAPIs, if required.

*Starting RFCs from
SAP ERP systems*

For Web AS ABAP and SAP ERP systems, the target system to be communicated with needs to be defined before RFCs can be made. This definition is made once and will be valid systemwide. It contains information on the logon procedure, the target system's IP address, and so on, and will be made via transaction SM59.

The RFC is an extension of the ABAP command `CALL FUNCTION`. When you add the target system that has been defined earlier, the function module will not be called in the Web AS ABAP or the SAP ERP system where the command is executed, but in the respective target system (see figure 3–2).

Figure 3–2
Function calls via RFC



*RFCs from proprietary
systems*

If an RFC is to be made from a proprietary system (or from a tool of a third-party supplier), an RFC library (RFC API) can be used. SAP provides RFC libraries for OS/2, Windows, Windows NT, and all UNIX derivatives released for ERP usage.

RFC connection types

An RFC can be made synchronously as well as asynchronously; it doesn't matter whether the called system is an SAP system or a customer-coded program.

Synchronous means that the call to the RFC component for the called system ends only when the remote function has been executed and terminated with a status code. On asynchronous calls, the status will not be directly returned to the triggering program; i.e., the call is technically terminated as soon as the data is “ready to be sent”. If the called system is not available at that time, in SAP BW the basis system will control further attempts to send.

A synchronous call is called transactional RFC (TRFC)⁸; otherwise, the term *RFC* is correct.

Transfer Tools

From a programming point of view (for SAP as well as for customers), the use of file and BAPI interfaces requires much effort if the programs are to use a wide range of communication options (synchronous, asynchronous, file, BAPI).

For this reason, a “toolkit” was built that provides a range of tools that require little programming effort, named *Application Link Enabling (ALE)*.

For the different communication techniques, ALE supports synchronous as well as asynchronous connections and provides monitoring and error correction functions.

For data communication, all information is transferred into so-called *intermediate documents (IDocs)*. This is a type of data container whose structure can be defined individually for the respective communication. Since the use of IDocs requires a given data model (header and data record), IDocs have a bigger overhead than, for example, a communication via RFC where no ALE/IDoc is used.

For this reason, the use of IDocs is unfavorable from a performance point of view. In BW, IDocs are only used to send requirements and confirmations (i.e., for transfers of low data volume).

Since they are so easy to program, IDocs were used for extraction of mass data in previous BW releases. Meanwhile, they have been replaced by RFCs in performance-critical areas, but sporadically they can be used optionally. Since they only serve the purpose of downward compatibility, there is no reason to use IDocs instead of RFCs in new development.



8. The TRFC was initially called asynchronous RFC. It was renamed transactional RFC since the asynchronous RFC has a different meaning in R/3.

3.1.2 Internet Communication Framework

There is a central infrastructure for Web AS ABAP and Web AS Java that enables communication via the HTTP, HTTPS, and SMTP protocols: the Internet Communication Framework (ICF) or the Internet Communication Manager (ICM), respectively.

Using ICF or ICM, BW can act as a server and offer HTTP services. Clients can send an HTTP request to BW. These requests are forwarded to the application via the ICM and from the ICM a reply can be returned to the client. This is the case with Web Reporting.

BW can also act as client and send requests to servers offering HTTP services and wait for their reply.

If required, HTTP services can be self-programmed. However, in practice this is rarely required since BW comes with a range of services that can be used both in the area of extraction and in the area of decision support systems. Here are some HTTP services:

- Accessing query data in HTML format via Web Analyzer
- Accessing query data in XML/A format via Open Analysis Interface
- Providing extraction data in XML format via the SOAP service (see chapter 2.1)
- Communicating between web pages and proprietary applications based on Business Server Pages (BSPs)
- Exchanging metadata between different systems in XML format (see appendix C.3)
- Showing metadata related to BW objects in HTML format
- Showing the user-defined documentation of BW objects

In the maintenance of services in the transaction SICF, you can see what services are offered and whether they are active (see figure 3–3).

The handling of a request via ICF is technically realized through a process in Web AS ABAP. There is an individual ABAP class behind each service, which in ICF terminology is referred to as *ICF handler* or *HTTP request handler*, respectively.

ICF handler

The name of the ICF handler can be found in the context menu of the respective service in the transaction SICF under the menu item *Display Service*→*Handler List*. For the HTTP Service for the Web Analyzer, this would be the ABAP class (= ICF Handler) CL_RSR_WWW_HTTP (see figure 3–4).

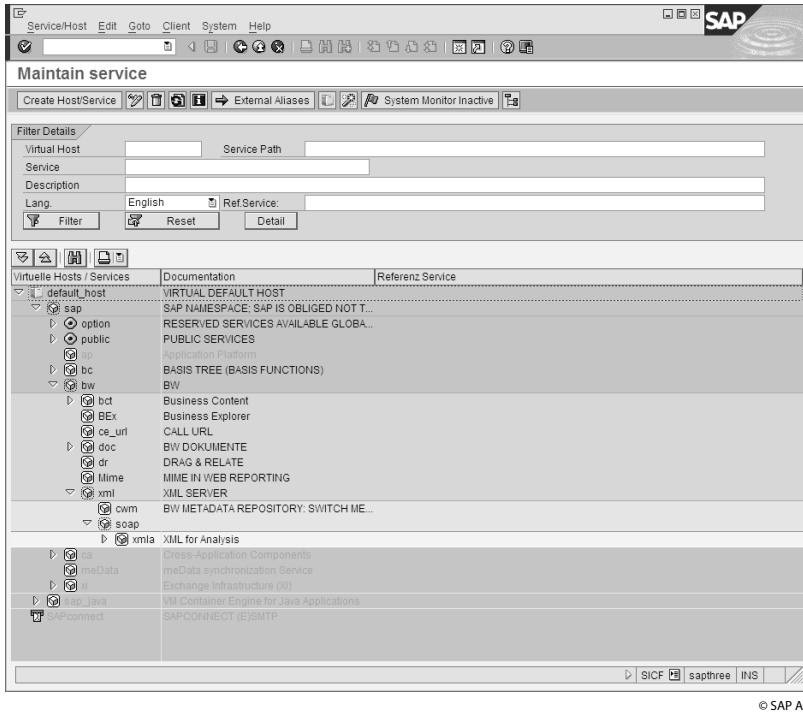


Figure 3-3
Services of the Internet
Connection Framework

© SAP AG

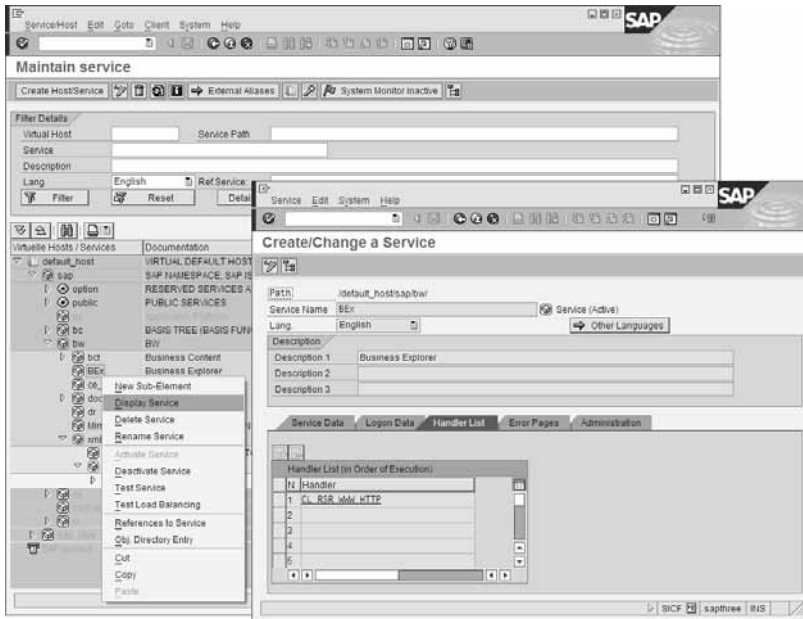


Figure 3-4
ICF services handler

© SAP AG

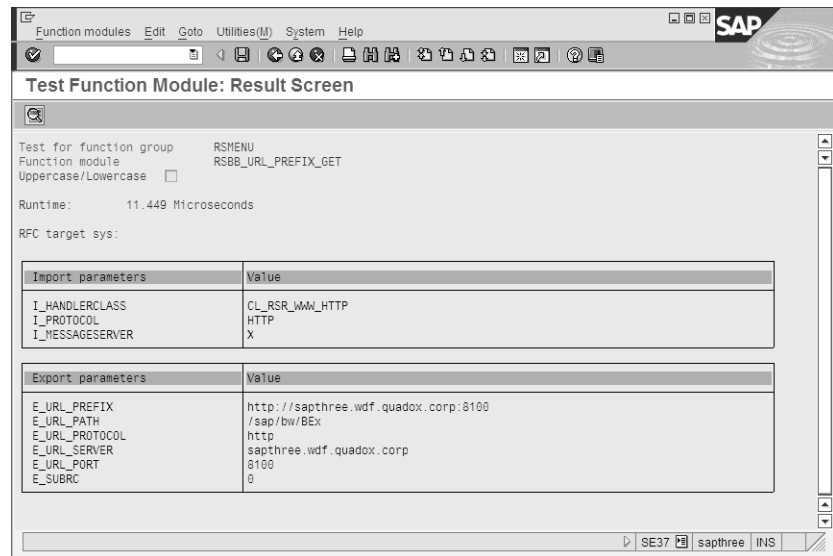
HTTP service launch The launch of an HTTP service requires a URL that identifies the server and the ICF handler, and that may transfer to the handler data that is coded within the URL as a query string.

If the service was supplied by SAP in BW, the URL of the HTTP service is always built according to the following schema:

<Protocol>://<Server>:<Port>/sap/bw/<Service>

With the use of the function module `RSBB_URL_PREFIX_GET`, the URL prefix can be identified. The function module can also be started manually in transaction SE37 (see the example of the URL prefix for the ICF Handler `CL_RSR_WW_HTTP` in figure 3–5).

Figure 3–5
Identification of the URL prefix
for ICF services



© SAP AG

Parameters to be indicated are the ICF handler (`I_HANDLER`) and one of the protocols, HTTP or HTTPS (`I_PROTOCOL`), and you need to specify whether the name of the message server (Parameter `X`) or the name of the application server (no parameter) or its alias should be coded into the URL.

When the function module is executed (menu item *Function Module* → *Test* → *Single Test* in transaction SE37), the respective ICF service must be active.

3.1.3 J2EE Connector Architecture

The J2EE Connector Architecture forms the basis for Web AS Java communication interfaces. This connector framework enables the connection from heterogenous systems to the J2EE platform on the one hand via **Java Connectors** and on the other hand via **Java Resource Adapters**.

Java Connectors are the middleware component (partly platform dependent) of an interface in which communication with the respective application system is implemented.

Java Connectors

The Java Resource Adapters connect Java Connectors and Web AS Java. Their interfaces are compliant with the standard interfaces of the J2EE Connector Architecture.

Java Resource Adapter

Thus, the Java Connectors or the Java Resource Adapter do not have to be supplied in Web AS Java, with delivery of SAP, but may be developed by the suppliers of other systems that are to be connected in the form of J2EE-compliant Java Resource Adapters and Java Connectors.

With the J2EE-compliant standard interface, it is possible to use the access to heterogenous systems in proprietary Java applications since they have access to the interfaces of the Java Resource Adapter.

An essential requirement for the use of the Web AS Java connectivity for BW (i.e., the Web AS ABAP) is communication between Web AS ABAP and Web AS Java.

*Communication between
Web AS ABAP and Web AS
Java*

For this purpose, there is a special Java Resource Adapter in Web AS Java: the SAP Java Resource Adapter (SAP JRA). The SAP Java Connector (SAP JCo) is used as a Java connector for this resource adapter; it is a middleware component that enabled communication between Java applications and the Web AS ABAP before Web AS Java was developed.

The SAP JCo supports the communication with the Web AS ABAP both ways (inbound and outbound); i.e., Web AS Java can start ABAP functions via an API and vice versa, and Web AS ABAP can call Java functions via an RFC connection. However, it has to be considered that SAP JRA can take RFC calls (via TCP/IP) but no TRFC calls, IDocs, and certificates.

SAP JCo and SAP JRA are self-contained software components that can be installed as stand-alone components too. However, SAP JRA is always installed as an add-on to SAP JCo, while SAP JCo is automatically installed together with Web AS Java.

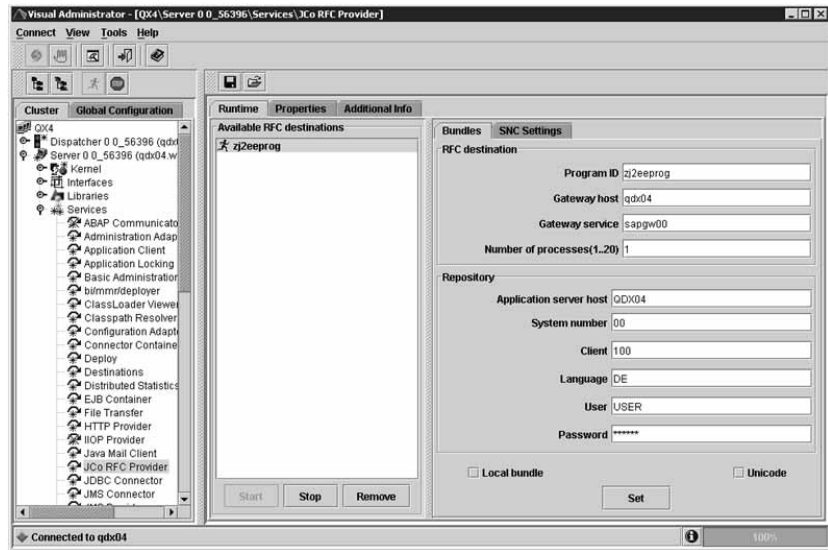
In principle, there is not always a Web AS ABAP for each Web AS Java. This means that each Web AS ABAP can basically communicate with

several (or no) Web AS Javas and vice versa. To configure the communication between both application platforms, you need to configure a **JCo RFC Provider** for Web AS Java and an **RFC Destination** for Web AS ABAP (for BW, the J2EE Engine is a system with an SAP-compliant interface).

Configuration of a JCo RFC
Providers

The configuration of the JCo RFC Provider is made in the Visual Administrator of the J2EE Engine under the tab *Cluster*→*Services*→*JCo RFC Provider* (see figure 3–6).

Figure 3–6
Configuration of the
JCo RFC Provider



With configuration of the JCo RFC Provider, a number of parameters have to be entered. In the case of the *RFC Destination*, which is to be provided by the J2EE server, these are the settings:

- **Program ID:** An arbitrary but unequivocal ID that clearly identifies the RFC Destination provided by the JCo RFC Provider (i.e., the RFC server program). The indicated program ID will later be used to configure the RFC connection in Web AS ABAP to address the JCo RFC Provider.
- **Gateway host:** The gateway server that processes requests to and from Web AS ABAP. If there is no dedicated gateway server, the Web AS ABAP will fulfill this role.
- **Gateway service:** The gateway service on a gateway server.
- **Number of processes:** The maximum number of running processes to be served via this RFC connection.

With the *Repository* specifications, you need to indicate the logon information for the J2EE server to log onto BW. This is the usual logon information comprising the application server host, the ERP system number, the client, and the language as well as the user and password.

With configuration of the JCo RFC Provider via the *Set* button, the specification regarding the RFC Destination is implicitly reviewed. If the gateway service can be reached by the indicated host, the JCo RFC Provider will be shown in the list of available RFC Destinations and can be started. When the service is started, the RFC Destination can be opened through Web AS ABAP.

You update the respective RFC Destination to call the JCo RFC Provider through Web AS ABAP. This is done with transaction SM59, and the destination must be defined as a new TCP/IP connection (Type T) (see figure 3–7).

Configure RFC Destination to J2EE Engine

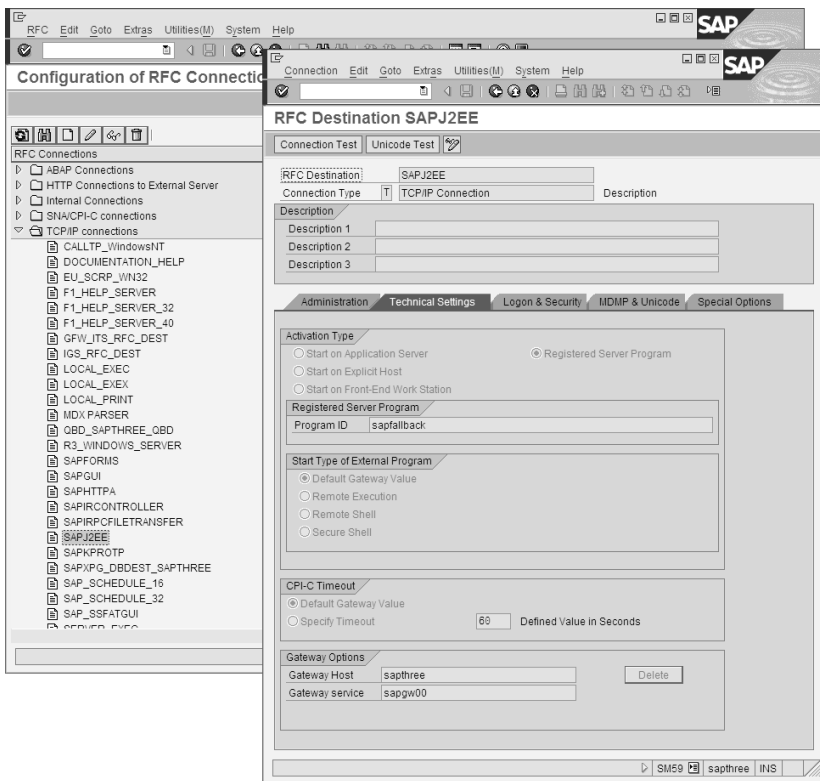


Figure 3–7
Configuration of the RFC Destination for the J2EE Engine

The start type needs to be the registered server program. For the program ID of the server program, provide the program ID under which the JCo RFC Provider within the J2EE Engine has been defined.

Choose the gateway host and service according to the JCo Provider specification.

4 Metadata Repository

Data warehouse systems will quickly turn into a “black box”⁹ if the applied data structures and data streams are not designed transparently. In such a system, correction of errors or modifications is disproportionately more complex and error-prone than in a transparently designed system.

BW responds to this issue with the Metadata Repository, where all objects that are related to the staging engine, the Data Manager, or BW’s own decision support tools are centrally stored.

The Metadata Repository usually contains the definitions of objects and settings directly related to the objects. Numerous settings that will be described in this book are not part of the specific object definitions and are not part of the Metadata Repository. Thus, do not assume that you can always find this information in the Metadata Repository.



The objects stored in the Metadata Repository usually fulfill different technical or business functions and can be clustered in three layers:

- Database
- ABAP Dictionary (formerly called Data Dictionary)
- Application (BW objects)

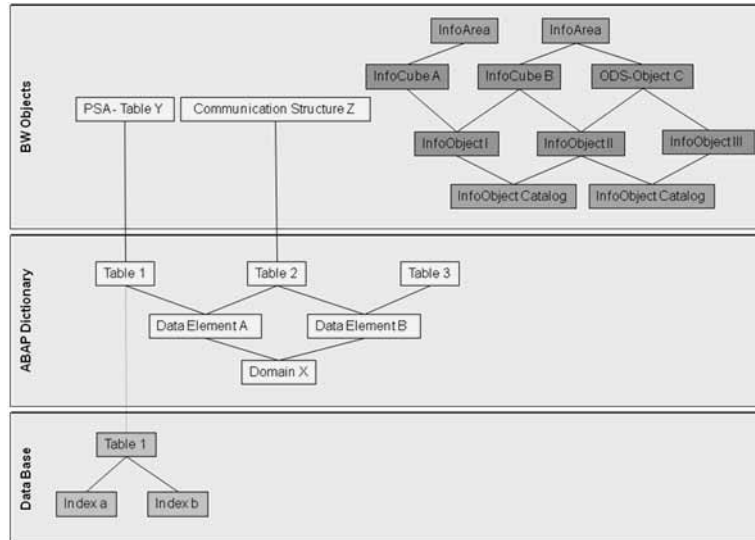
Figure 4–1 shows the three layers in an sample selection of typical objects.

To work in BW, it is crucial that you understand the individual object layers. In the following sections, we will characterize the layers and explain their function within the Metadata Repository.

Please note that the three layers are not administered in a joint Metadata Repository but each in its own Repository.

9. This means a system that processes and stores data in an incomprehensible way and thus is similar to a black box, where the interior is invisible and remains unknown.

Figure 4-1
BW object layers



4.1 Database Objects

The physical data storage is done in the database system. There, data is filed with the help of database objects (tables, indexes).

The stored data may be of temporary or persistent nature and may assume different tasks. The database system has its own metadata on the stored structures, but in the case of SAP BW, it has no knowledge of the semantics or business context of the data.

There are two main disadvantages to directly accessing objects of a database system from BW programs:

- Direct access to the database objects requires that access to applications are designed depending on the database system used. This does not make sense from an administration perspective. It would also require comprehensive database-specific knowledge on the application programmers' part.
- The database does not provide any semantic information on the data structures and is thus not clear enough for application development.

Due to these disadvantages, the database objects are isolated from direct access by the BW programs. Instead, BW programs have to access the objects in the ABAP Dictionary.

4.2 ABAP Dictionary Objects

The ABAP Dictionary is a form of Metadata Repository for the BW basis system's data structures that apply to all applications system wide. The BW-relevant basic elements are as follows:

- Domains
- Data elements
- Tables

As far as these elements can be displayed in tables in the database, BW programs can access the ABAP Dictionary with the help of a standardized Structured Query Language (SQL) dialect, *Open SQL*. When the ABAP Dictionary is accessed, the basis system translates the Open SQL commands to database-specific commands. Thus, all applications based on the basis system (i.e., all BW programs) are database independent.¹⁰

The elements of the ABAP Dictionary do not necessarily have to be duplicated in the database. This is the case when the ABAP Dictionary elements contain metadata that does not show any data content (e.g., field structures).

Using transaction SE11, you can display, change, or create the elements of the ABAP Dictionary (see figure 4–2).

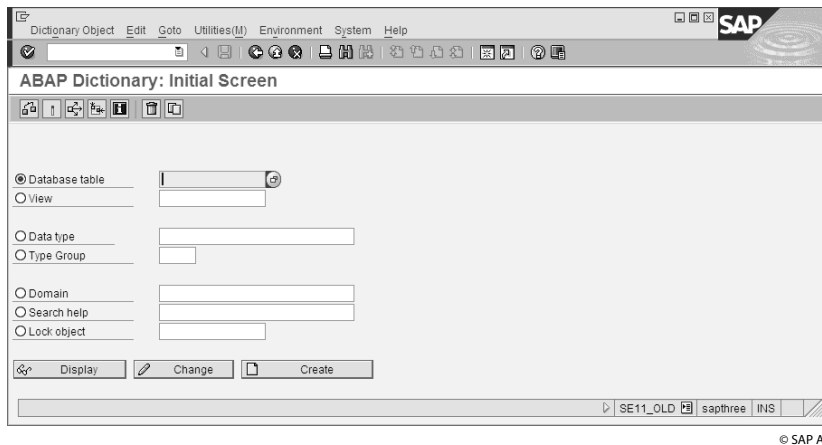


Figure 4–2
Starting the ABAP Dictionary

10. In some areas, BW uses special features of individual database systems and this requires particular handling by the ABAP programs depending on the corresponding platform. If necessary, the book will advise accordingly.

In the following sections, the basic elements of the ABAP Dictionary will be explained. In section 4.2.4, the concept of development namespaces will be discussed.

4.2.1 Domains

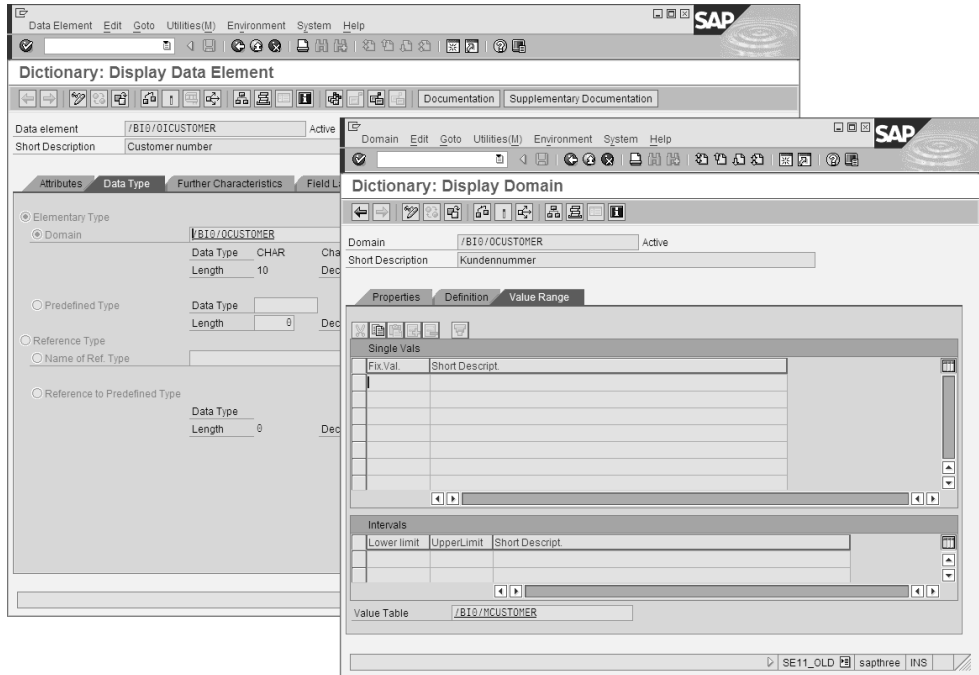
Domains define the basic business elements of the ABAP Dictionary using a data type and a data length (e.g., customer number, material text, turnover key figure).

For example, a customer number is defined as a 10-digit alphanumeric field by the domain `/B10/OCUSTOMER` with the data type `CHAR` of length 10. All further ABAP Dictionary elements will be based on *domains*.

A particular feature in the definition of domains is the option to store the reference in a check table. Typically, this is a master data table where all values are stored that the domain duplicates in a master data table.

Check tables are especially valuable when entries are validated or the user chooses a value from a selection. Figure 4–3 shows the definition of the domain `/B10/OCUSTOMER` (customer number) as `CHAR` with length 10 and the value table `/B10/MCUSTOMER`.

Figure 4–3
Domains in the ABAP
Dictionary



4.2.2 Data Elements

Data elements are the describing layer over the domains. Each data element refers specifically to a domain, and each domain can be used by several data elements.

A data element contains descriptive data that can be used as a header when displaying field content. Data elements serve to generate a description from the general definition of a basic business element (the domain) that meets the needs of the specific use of the domain.

An example is the domain *customer number*. In sales, the customer number is never used as customer number but as one of the partner roles *customer*, *regulator*, or *supplier*. To accomplish this use of the domain, three data elements can be defined (customer, regulator, supplier), where each refers to the domain *customer number*.

Figure 4–4 shows the definition of the customer as a data element based on the domain *customer number*.

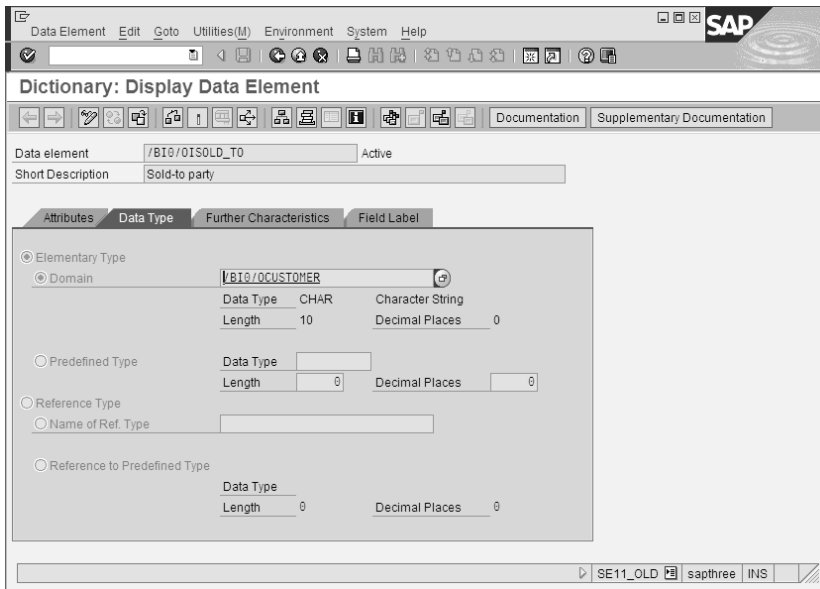


Figure 4–4
Data elements in the ABAP Dictionary

4.2.3 Tables

In the ABAP Dictionary, a structure of several fields is defined in a table. All fields of the structure need to be defined by a data element. Only the field name will be defined in the table. Within the table, the field name

needs to be unambiguous. Within a table, a data element can be used repeatedly.

A table is used to bring complex business information into context. One example is the definition of a table to display the customer master data in a combination of customer number, name, phone number, etc.

There are two table types in the ABAP Dictionary:

- Transparent tables
- Structures

Transparent tables

Transparent tables are created in the ABAP Dictionary as well as in the database (the basis system does this automatically during table maintenance). Therefore, transparent tables offer an option to store data in the database system.

Any data stored through BW applications will be stored in database tables that are defined as transparent tables in the ABAP Dictionary.

Structures

Similar to data elements and domains, structures are components of the ABAP Dictionary. Structures can be reused in programs as preconfigured type declarations. BW intensely uses structures for data flow definition.

The advantage of structures is that they are maintained in one central place, the ABAP Dictionary, and any program having to this structure will be able to experience and use any change to the structure.

4.2.4 Development Namespaces

To avoid naming conflicts between proprietary ABAP Dictionary objects and ABAP Dictionary objects supplied by SAP, there are special ***development namespaces*** for all ABAP Dictionary objects. All customer and SAP-supplied objects have to adhere to the naming conventions.

SAP namespace

Thus, all SAP-supplied objects start with the letters A to X.

For ABAP Dictionary elements that are not SAP supplied but generated in SAP standard programs, the namespace `/Bxx/` has been reserved, where x stands for the letters A to Z.

For SAP BW, the namespaces `/BIO/` and `/BIC/` are especially important; they are created during metadata maintenance of BW content objects and customized BW objects (see section 4.3). Thus, the ABAP Dictionary objects related to customized BW objects are also filed in the SAP namespace.

Any customized ABAP Dictionary object starts with Y or Z, but this only applies to objects that are created in the ABAP Dictionary directly.

ABAP Dictionary objects that are generated by BW based on the meta-data layer of the BW objects are not stored in the customer namespace even if the respective BW object is a customer-made development.

The definition of SAP and customer namespaces ensures that proprietary development from SAP users do not overlap with the supplied standard programs.

Within the customer namespace, it is up to each customer to agree on the use of the namespace for the object development in their organization.

System vendors, however, provide objects where neither the SAP namespace (this would interfere with SAP) nor the customer namespace (this would interfere with the customer) can be used.

Corporations may also face this problem when development is done centrally and locally, e.g., central development from headquarters and country-specific enhancements to globally used development systems.

For this reason, ABAP Dictionary objects can be stored in a so-called customer and partner namespace that is allocated by SAP.¹¹ It is marked by the prefix */xxxxx/*, where *xxxxx* stands for a 5- (minimum) to 10-digit (maximum) alphanumeric string, starting with a letter, that refers to the respective company (e.g., */QUADOX/*).

You request such a customer or partner namespace specifically for selected installations to ensure that the SAP-provided namespace can be used globally on the respective development systems.

Standard customer namespace

Own customer namespace

4.3 BW Objects

While the Repository elements of the ABAP Dictionary already supply simple semantic information on the stored data and data structures, there are other objects in BW that define all application functions, from the data model and the data flow to monitoring. And for this purpose, they require more detailed semantic information. Here are some samples for such BW objects:

- *InfoObjects* to define business characteristics and key figures
- *BasisCubes* to take analysis-relevant transactional data

11. Development namespaces can be requested from SAP free of charge. Details on the application and the installation of namespaces can be found on the SAP Service Marketplace under the reference number 84282.

- *InfoAreas* to group InfoObjects and BasisCubes
- *Transformation rules* to define the data flow
- *Queries* to make analyses

The individual BW objects will be explained in detail in the following sections. To better clarify the tasks of BW objects, we will use the master data InfoObject, which represents business reference parameters (e.g., customers, products), as a sample for BW objects. The InfoObject disposes of master data that needs to be stored in several database tables.

With the InfoObject, information on its reporting behavior is stored. This is not only descriptive information (where a data element would be sufficient), but also information on its aggregation behavior, geocoding, etc. This information will also be stored in tables in the database system.

For this reason, BW objects are complex structures that are created on the Data Warehousing Workbench level with individual characteristics and from which a whole range of objects are usually generated in the ABAP Dictionary.

To make the development of BW objects easier and more transparent, the metadata on each BW object is stored in different versions: active, modified, or delivered.

*A version*⁴

The A version of metadata on BW objects corresponds with the objects generated for the BW object in the ABAP Dictionary and represents the actively used form of metadata.

*M version*⁵

With a redevelopment or modification of BW objects, the changes (e.g., adding fields to the master data) will not immediately transfer into the objects generated in the ABAP Dictionary and the programs; instead, they will first be stored as metadata in an M version. This version exists in addition to the A version.

Only when the metadata is activated with the modeling of the BW objects will the changes be transferred from the M to the A version, and then the objects of the ABAP Dictionary and the programs will be generated from the metadata definitions.

*D version*⁶

BW objects that come with a content system¹⁵ are—with their delivery—not created as an active version in the metadata and the ABAP

12. A = Active (active version)

13. M = Modified (modified version)

14. D = Delivered (delivered version of BI content)

15. This refers to the BI content of SAP as well as the customer and partner content that has been provided by proprietary content systems. More detailed information on content development can be found in appendix C.2.

Dictionary. Instead, the metadata of these objects will be stored in a D version. Only when the BW objects of the metadata content are transferred (see appendix C.1.5), will the content of the D version be transferred into the A version and respective objects generated in the ABAP Dictionary and the programs.

The D version will not be deleted when installed, so it can be transferred several times, and transferred BW objects can be modified or deleted without any consequences to the D version.

As described when we explained architecture components, the Data Warehousing Workbench in BW is used as central tool to define metadata. Further, it offers research options for the Metadata Repository (see figure 4–5).

Metadata research

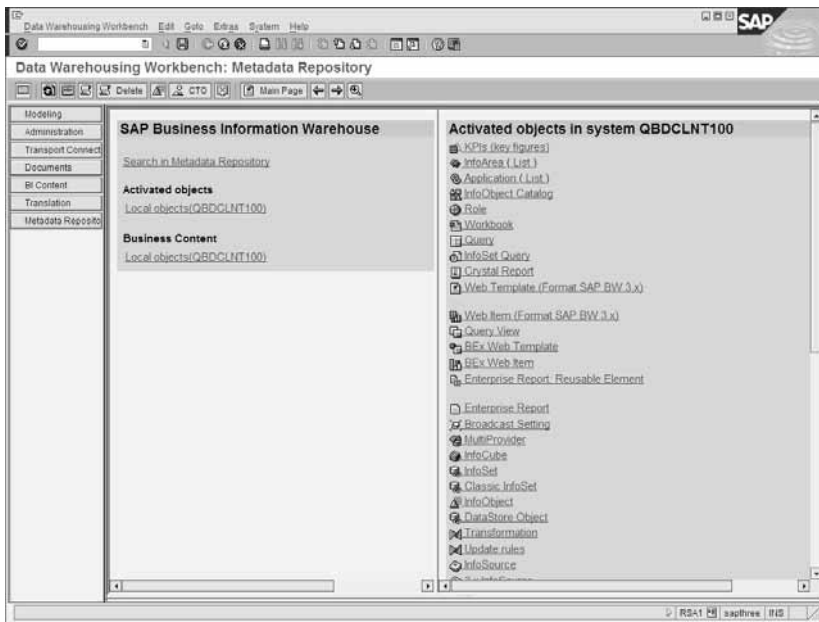


Figure 4–5

The Metadata Repository

The search function excludes metadata objects of the ABAP Dictionary as well as metadata of the database system. Also, the search function does not consider any object-independent settings. Thus, in a lot of cases, the Metadata Repository of the Data Warehousing Workbench is rather unqualified to give a comprehensive overview on the operation method of the system.

4.3.1 BW Namespaces

BW objects are defined on an application level so that the development namespaces of the ABAP dictionary do not apply (see section 4.2.4). BW's own namespaces are applicable.

BW default namespace

BW default namespaces are used to avoid conflicts between BW objects from SAP (e.g., BI content) and user-defined BW objects.

All BW objects from SAP are marked with the prefix 0 to 9 in the BW default namespace; that is, the name of each BW object that is supplied by SAP or generated by the system will begin with a number from 0 to 9. The name of each customer-defined BW object will have a prefix from A to Z.

For objects of the ABAP Dictionary that are generated from metadata of BW objects, there are **generation namespaces** that apply to the respective BW default namespaces. This is the generation namespace /B10/ for BW objects from SAP and /BIC/ for customer-defined BW objects.

This means, for example, that the InfoObject 0CUSTOMER from the BI content generates a master data table in the ABAP Dictionary with the name /B10/PCUSTOMER. However, the customer-defined InfoObject ZCUSTOMER generates in the ABAP Dictionary a master data table named /BIC/PZCUSTOMER.

BW partner namespace

As with development namespaces in the ABAP Dictionary, system vendors and customers with central development systems may refer to their own customer/partner namespaces for the definition of BW objects.



Proprietary customer/partner namespaces are mostly used by system vendors. For corporations with a global IT landscape, they are hardly relevant. Do not be worried if you do not see the need for your own partner namespace in your company. You can keep developing in the BW default namespaces without hesitation.

For this purpose, a development namespace for BW objects as well as a generation namespace has to be requested from SAP and set up in the system (see section 4.2.4).

While the development namespace for BW objects follows regular development namespace conventions, different rules apply to the generation namespace. The name has to be encased with slashes (/) and must not exceed seven characters (slashes included). Also, the namespace has to start with B and otherwise must contain only numbers. A valid generation namespace would be /B10/.

In the transaction RSNSPACE (see figure 4–6), the generation namespace will be assigned to the BW partner namespace and it will apply to the objects in the ABAP Dictionary that are deduced from the metadata of the BW objects.

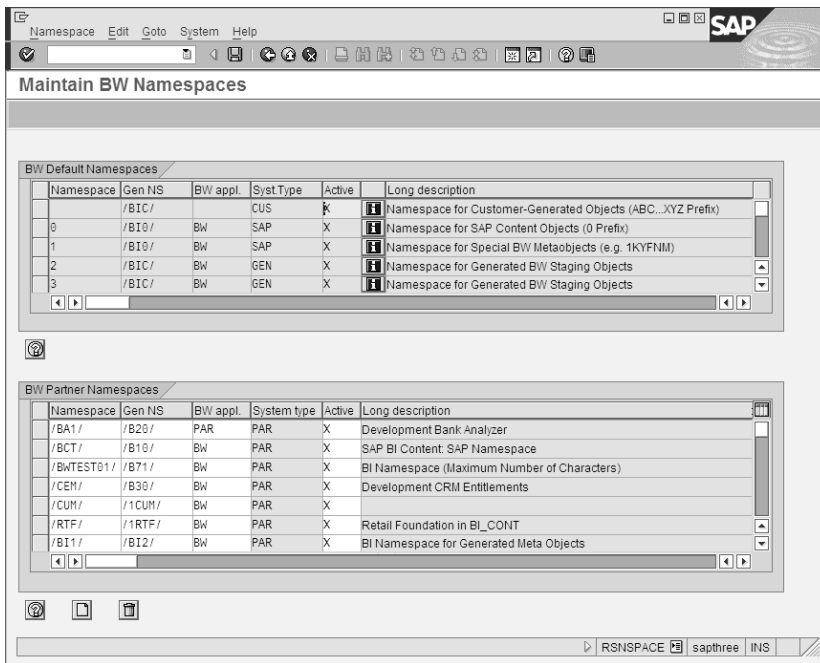


Figure 4–6
BW namespace maintenance

According to the namespaces defined in the figure, the InfoObject /BQDX/CUST will generate a master data table named /B10/PCUST in the ABAP Dictionary.