

An Overview on Data Dictionary



Applies to:

SAP NetWeaver 7.0, ABAP, to all those who wants to learn about Data Dictionary. For more information, visit the [ABAP homepage](#).

Summary

This article will give the overview of ABAP dictionary and predefined data elements.

Author: Tanmaya Gupta

Company: Infosys tech. ltd

Created on: 1 May 2010

Author Bio



Tanmaya Gupta is working as a system engineer for Infosys technologies limited from past 1.6 years. He has an experience of 1years in SAP ABAP. He loves to share his knowledge to his fellow mates and to help others whenever required.

Table of Contents

Data Dictionary	3
Objects in Data Dictionary	4
Tables:	4
Views:	4
Types:	4
Domains:.....	4
Search Helps:	5
Lock objects:.....	5
Data Types	5
Built in Elementary Data Types	6
Mapping of the Data Types	7
User-Defined Data Types	10
Elementary Data Types	10
Reference Data Types	10
Complex Data Types	10
Examples of Complex Data Types:	10
Related Content	12
Copyright	13

Data Dictionary

Data Dictionary, also known as *'metadata repository'*, as defined by 'IBM Dictionary of Computing' is a "central repository of information about data such as meaning, relationship to other data, origin, usage, and format". ABAP Dictionary centrally describes and manages all the data definitions used in the system and the database. It is completely integrated in the ABAP Development Workbench. All the other components present in the workbench actively access the definitions stored in the data dictionary. ABAP Dictionary supports the definition of user-defined types i.e. data elements, structures and table types. These types are used in the ABAP processors and ABAP programs. It also defines the structure of database objects i.e. tables, views and indexes. These database objects automatically get created in the underlying database with the definition of data dictionary when the objects are activated. It also provides editing tools like Search help and locking tool such as lock objects. Thus, objects present in the ABAP Dictionary are tables, views, types (data elements, structures and table types), domain, search helps and lock objects.

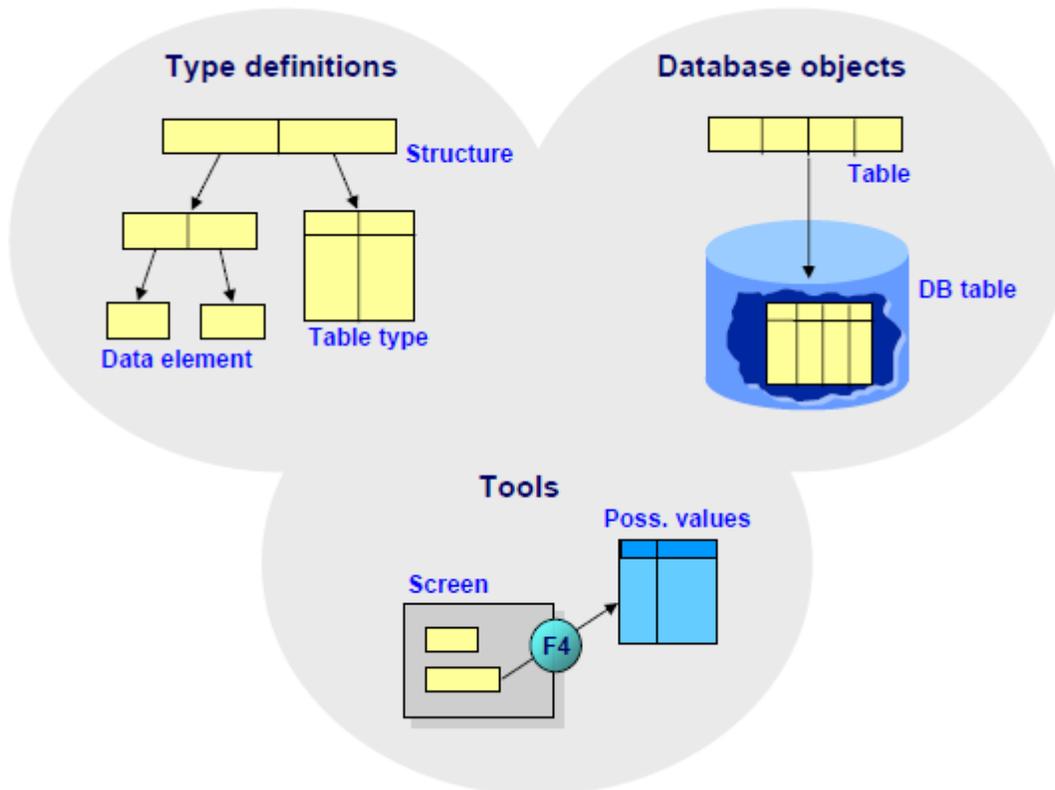


Fig 1: Objects under Data Dictionary

The ABAP Dictionary with the help of their objects ensures the following:

- ✚ Enforces Data integrity
- ✚ Manages data definitions without redundancy
- ✚ Tightly integrated with rest of the ABAP/4 Development Workbench.

Enforcing data integrity ensures that data entered into the system is logical, complete and consistent. Because of data integrity rules, system automatically prevents the entry of invalid data. Defining the data integrity rules at the dictionary level means they only have to be defined once, rather than in each program that accesses that data.

Examples of lack of data integrity:

A 'date' field with more than 31 days.

Order assigned to the customer with invalid customer number.

Managing data definitions without redundancy is the process of linking information to their corresponding data elements. For example, an equipment master table will contain the equipment number in several places and several tables. The definition of equipment number is defined in only one central place. This central definition is defined for each instances of the equipment number, wherever it is used.

The ABAP Dictionary's is integrated to rest of the ABAP workbench. Because of this reason, ABAP programs automatically recognize the name and characteristics of dictionary objects. System also provides the easy navigation between the ABAP objects and dictionary objects. If you double-click on the dictionary object from the program code, system will take you to the definition of that object in the ABAP/4 Dictionary. Whenever the dictionary object is changed the program code referring to that object will always refer to the new version of that dictionary object. Since ABAP is an interpreted language, its interpreter sees only internal representation of data dictionary objects. These internal representations are adjusted automatically when the system finds that changes have been made in the ABAP Dictionary. Because of this adjustment, screens, input help, database interface, and development tools always access current data. Inactive ABAP Dictionary objects have no effect on the runtime system. Inactive objects can also be activated together when all the changes are done.

For Example:

```
Data: equi_1 type table of equi.
Select *
from equi
into table equi_1.
```

Above ABAP code will declare internal table 'equi_1' as a table type of 'equi'. All the entries of 'equi' table are selected and copied to internal table 'equi_1'. Here in program only internal table 'equi_1' is declared but all other information like field names, data types and field lengths, are copied from the table 'EQUI', which is defined in the ABAP Dictionary.

Objects in Data Dictionary

Objects created in the ABAP Dictionary are created in the underlying relational database using these data definitions. The ABAP dictionary thus describes the logical structure of the object used in the application development and shows how they are mapped to the underlying relational database in table or view. As discussed, most important object types in the ABAP dictionary are tables, views, types, domains, search helps and lock objects.

Tables:

Tables are defined independently of the database. First the table structure is created in the ABAP dictionary then the table is created in the underlying database from this structure.

Views:

Views combines more than one table. The structure of the view is defined in the ABAP Dictionary. With the help of views, application-dependent view can be defined that combines the data.

Types:

Types are generally used in ABAP programs and ABAP dictionary to define the structure of the fields, variables, constants, tables, etc. Types can be further classified as elementary types, reference types or complex types. Data element, structure and table types are the type categories under user defined data types.

Domains:

Domain is a central object which defines the technical attribute of a field. Technical attribute refers to data type, length, fixed value and interval of a field (field of the table, structure, table type etc.). Domain is assigned to the data element which in turn is assigned to the table fields or structure fields. So all the fields which use the data element have the technical settings defined by the domain.

Search Helps:

With search help user can display the list of all possible input values for a screen field. Whenever 'F4' key is pressed on the screen field, popup will appear having the list of entries which satisfies the search in that screen field. You can select the needed record. There are three types of search help, 'elementary search helps', 'collective search helps' and 'Append search helps'.

Lock objects:

In R/3 system, multiple users can access same object simultaneously. When one person is editing the program/transaction, locks are set so that no other person can make changes to that program at the same time. Function modules are generated automatically from the lock object definition of ABAP Dictionary. These function modules are responsible to enqueue or dequeue the locks on the objects.

Data Types

Global data types can be defined with the ABAP Dictionary. These global data types can be use in ABAP program by TYPE command. The data types of database tables are a subset of all possible types, namely flat structures. Thus, data type in the ABAP Dictionary is the user's view on the data. ABAP Data Types can be divided into two categories.

- ❖ Build-In elementary Data Types
- ❖ User-Defined Data Types

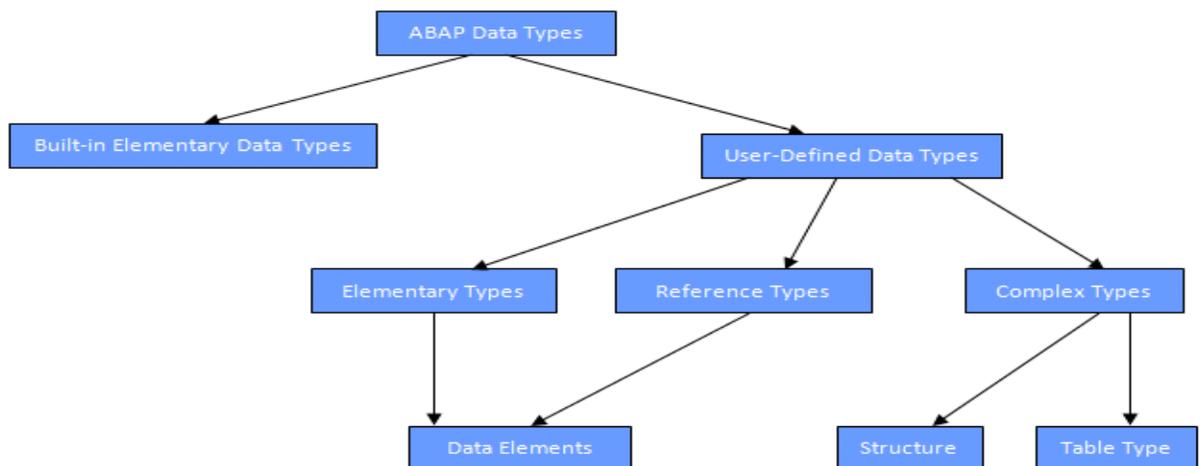


Fig 2: Data Types in ABAP Dictionary

Built in Elementary Data Types

There are some Data types which are already defined in ABAP Dictionary. These data types are converted into their corresponding ABAP Data type, during runtime. Some data types have a predefined length and set templates for output.

Existing Data Types are:

- **ACCP**: Posting period. The length of this data type is set to 6 places and the format is YYYYMM. In input and output, a point '.' is inserted between the year and month, so the template of this data type has the form '____.____'. For Ex: Posting period '201004' will be displayed as '2010.04'.
- **CHAR**: Character string. This data type can have maximum length of only 255 in tables. If longer character fields are to be used in tables, data type LCHR should be used. There are no restrictions on the length of such fields in structures. For Ex: if we want to represent any string such as 'Name', 'Address' etc., we have to use data type CHAR.
- **CLNT**: Client. Client fields always have three places. It is used in objects to make that object client dependent. If this field is not used in the object then that object will be considered as client independent and can be used in all the clients. For Example: Client '100', Client '200' etc.
- **CUKY**: Currency key. Fields of this type are referenced by fields of type CURR. The length is set to 5 places for this data type.
- **CURR**: Currency field. Equivalent to an amount field DEC. A field of this type must refer to a field of type CUKY (reference field). The maximum length for this data type is 31 places.
- **DATS**: Date. The length of this data type is 8 places and the output length can be defined with the user profile. Date is represented in YYYYMMDD format internally in database. For Ex: Date 24th April 2010, will appear as 20100424 internally and as 2010/04/24 if the date is separated by '/' in user's output format.
- **DEC**: Decimal. This data type represents the amount field with decimal places, sign and commas separating thousand. It has a maximum length of 31 places. For ex: Number '-12345' will be represented by this data field as -12,345 when displayed.
- **FLTP**: Floating point number. This data type is used to represent floating point numbers. It can have maximum length of 16 places including decimal places.
- **INT1**: It is a one byte integer which can have values between 0 to 255. The length is set to 3 places for this data type.
- **INT2**: It is a 2-byte integer which can have value between $-32767 (-2^{16-1} - 1)$ and $32767(2^{16-1} - 1)$. This field should only be used for long fields. These long fields are positioned immediately in front of a long field (type LCHR, LRAW). With INSERT or UPDATE on the long field, the database interface enters the length which was actually used in the length field. The length is set to 5 places for this data type.
- **INT4**: 4-byte integer between $-2147483647(-2^{32-1} - 1)$ and $2147483647(2^{32-1} - 1)$. The length is set to 10 places for this data type.
- **LANG**: Language key. Has its own field format for special functions. This data type always has length 1. The language key is displayed at the user interface with 2 places, but is only stored with 1 place in the database. The conversion exit ISOLA converts the display at the user interface for the database and vice versa. This conversion exit is automatically allocated to a domain with data type LANG at activation.
- **LCHR**: Character string of any length, but with a minimum of 256 characters. Fields of this type must be located at the end of transparent tables and must be preceded by a length field of type INT2. If there is an INSERT or UPDATE in ABAP programs, this length field must be filled with the length actually required. A field of this type cannot be used in the WHERE condition of a SELECT statement.
- **LRAW**: Uninterpreted byte string of any length, but with a minimum length of 256. Fields of this type must be located at the end of transparent tables and must be preceded by a length field of type INT2. If there is an INSERT or UPDATE in ABAP programs, this length field must be filled with the length actually required. A field of this type cannot be used in the WHERE condition of a SELECT statement.

- **NUMC**: Long character field in which only numbers can be entered. The length of this field is limited to a maximum of 255 places.
- **PREC**: Accuracy of a QUAN field. The length is set to 2 places for this data type.
- **QUAN**: Quantity. Equivalent to an amount field DEC. A field of this type must always refer to a unit field with UNIT format (reference field). The maximum length for this data type is 31 places.
- **RAW**: Uninterpreted byte string. Fields of type RAW may have only a maximum length of 255 in tables. If longer raw fields are required in tables, you should select data type LRAW.
- **RAWSTRING**: Uninterpreted byte string of variable length This type can only be used in types (data elements, structures, table types) and domains. It cannot be used in database tables. In ABAP, this type is implemented as a reference to a storage area of variable size.
- **STRING**: Character string with variable length. This data type can only be used in types (data elements, structures, table types) and domains. It cannot be used in database tables. In ABAP, this type is implemented as a reference to a storage area of variable size.
- **TIMS**: Time. The length is set to 6 places for this data type. The format is hhmmss. The template for input and output has the form '__.__.____'. For Example: '103045' will be displayed as '10.30.45'.
- **UNIT**: Unit. Fields of this type are referenced by fields of type QUAN. The length of this data type is set to 2 or 3 places.
- **VARC**: Character field of variable length. Creation of new fields of this data type is not supported as of Release 3.0. However, existing fields with this data type can still be used. A field of this type cannot be used in the WHERE condition of a SELECT statement.

Note: In Numeric data types CURR, DEC, FLTP, INT2, INT4 and QUAN, you can choose whether or not a sign should be displayed on screens. With the data types CURR, DEC, QUAN, comma separated thousands and decimal points are set by the system. In data type DATS, ACCP, TIMS, decimal point '.' is set automatically by the system. The output length (number of places and the number of necessary editing character such as comma separated thousand and decimal points) is greater than the specified length. System automatically calculates output length when defined in domain. However, if the length specified in domain is less than the calculated output length, system will omit the editing characters.

Mapping of the Data Types

ABAP Dictionary and ABAP programming language have different data types. ABAP Dictionary has more predefined types than the ABAP Programming language. The data types are different because the predefined data types in the ABAP Dictionary have to be compatible with the external data types of the database tables supported by R/3. The ABAP processor uses the ABAP data types in the programs to define the variables, constants, work area of tables, structures etc.

ABAP data types used in ABAP Processor are as follows:

- ❖ **C**: Character. This data type is used to represent character types.
- ❖ **D**: Date, format YYYYMMDD
- ❖ **F**: Floating point number in DOUBLE PRECISION (8 bytes)
- ❖ **I**: Integer
- ❖ **N**: Numeric character string of arbitrary length
- ❖ **P**: Amount or counter field (packed; implementation depends on hardware platform)
- ❖ **S**: Time stamp YYYYMMDDHHMMSS
- ❖ **T**: Time of day HHMMSS
- ❖ **V**: Character string of variable length, length is given in the first two bytes
- ❖ **X**: Hexadecimal (binary) storage
- ❖ **STRING**: Character string of variable length
- ❖ **XSTRING**: Uninterpreted byte string of variable length

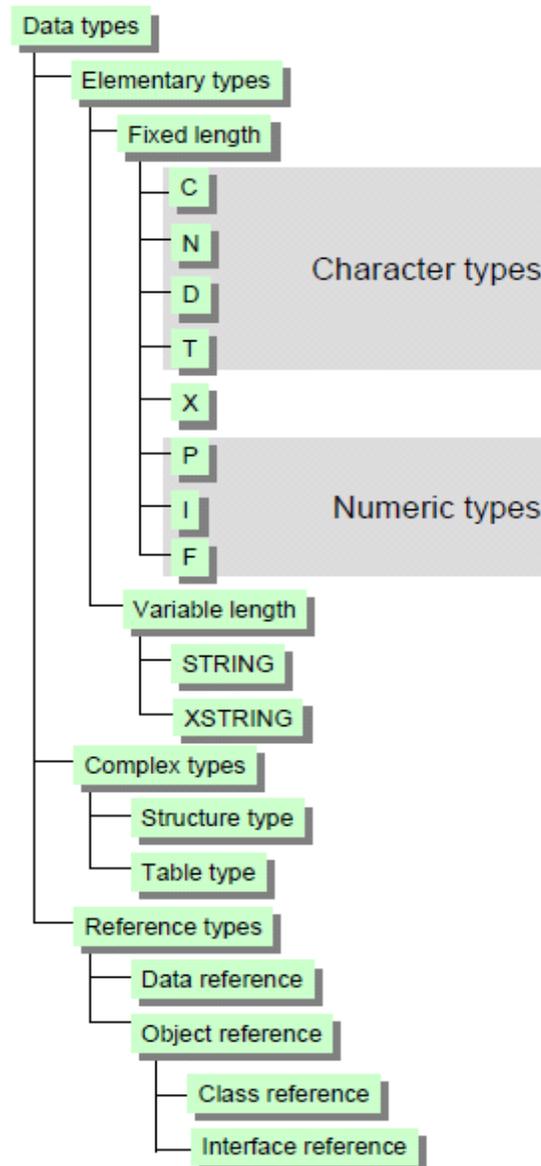


Fig 3: Data types in ABAP programming language

If a data element or a field of an ABAP Dictionary object (structure, table type, table and view) is used in an ABAP program, the Dictionary data type is converted to the corresponding ABAP data type. Let us see, how ABAP dictionary types and ABAP processor data types are mapped.

Dictionary Type	Meaning	Maximum Length n	ABAP type
ACCP	Accounting period YYYYMM	6	N(6)
CHAR n	Character	1-255	C(n)
CLNT	Client	3	C(3)
CUKY	Currency key	5	C(5)
CURR n, m	Currency field	1-17	P((n+1)/2) DECIMAL m
DEC n, m	Calculation/amount	1-31, 1-17 in	P((n+1)/2)

	field	tables	DECIMAL m
DATS	Date	8	D(8)
FLTP	Floating point number	16	F(8)
INT1	Single-byte integer	3	X(1), Internal only
INT2	Two-byte integer	5	X(2), Internal only
INT4	Four-byte integer	10	X(4), Internal only
LANG	Language	Internal 1, external 2	C(1)
NUMC n	Numeric text	1-255	N(n)
PREC	Accuracy	2	X(2)
QUAN n, m	Amount	1-17	P((n+1)/2) DECIMAL m
RAW n	Byte sequence	1-255	X(n)
TIMS	Time HHMMSS	6	T(6)
UNIT	Unit	2-3	C(n)
LRAW	Long byte sequence	256-max	X(n)
LCHR	Long character	256-max	C(n)
STRING	String of variable length	1-max	STRING
RAWSTRING	Byte sequence of variable length	1-max	XSTRING

The characters used in the above table mean:

n: number of places of the field in the ABAP Dictionary

m: number of decimal places of the field in the ABAP Dictionary

'max' in LCHR and LRAW is the value of preceding INT2 field. The 'internal' length of the LANG fields is in the dictionary, the 'external' length refers to the display on the screen.

User-Defined Data Types

User defined data types can be defined and stored for all the programs in the ABAP Dictionary. These user defined data types can be defined either globally in ABAP Dictionary or locally in ABAP programs with TYPES command. Both the declarations provide the same functionalities which define the user defined types according to the business and program requirement. The types defined globally in the Data Dictionary can be accessed by ABAP programs to define data objects.

For Ex: suppose, we want to declare a variable 'NAME' which is character type of length 30. We can do so either locally in program or we can create a data element and use that data element in program.

'Data Name(30) type c.'. This command will declare the variable locally in the ABAP program.

'Data Name type char30.'. Here, char30 represents the data element which is referred to the domain 'CHAR30' having 'data type' as CHAR and 'No. Characters' as 30.

User defined data types can be further classified into three categories.

- Elementary Data Types
- Reference Data Types
- Complex data types

Elementary Data Types

Elementary types are part of the dual-level domain concept for fields in the ABAP Dictionary. The elementary type has semantic attribute such as data type, length, texts, value tables, documentation etc.

Elementary types are described by the Data Elements. Data type can be specified in two ways.

- **By directly assigning to an ABAP dictionary type.**
You can directly assign a predefined ABAP Dictionary type and a number of characters to an elementary type.
- **Assigning to a domain.**
The technical attributes are inherited from the domain. Domain specifies the technical specification of the data element.

Reference Data Types

Reference types describe the data objects that contain references (pointers) to other objects (data objects and objects in ABAP Objects). However there exist no predefined references- you have to define them explicitly. There is a hierarchy of the reference types that describes the hierarchy of objects to which references can point.

Complex Data Types

Complex types are made up of other types. You can access a complex data types either as a whole or by the individual component. Complex data types group semantically related data under the single name and manage and process them. There are no predefined complex data types defined in ABAP. They can either be defined in ABAP program or in the ABAP Dictionary.

Examples of Complex Data Types:

Let us consider some examples of complex data types which are arranged in ascending order of complexity.

1. Structures consisting of a series of elementary data types of fixed length.
2. An internal table whose line type is an elementary type.
3. Internal table whose line type is a non-nested structure.
4. Structure with structures as components.
5. Structures containing internal table as components.
6. Internal table whose line type contains further internal tables.

The following type categories can be defined in the ABAP Dictionary:

- **Data Elements:** Data element describes either an *elementary type* or *reference type*. Data elements are used to define the type of the table field, structure component or the row type of a table type. All the semantic information about the meaning of the table field or structure component and information about editing the corresponding screen field can be assigned to a data element
- **Structures:** Structures defines the complex types. *Structure types* describe the structure and functions of any structured data objects that is data structures with components of any type. Thus, a component can be a field of elementary type, reference type or structure type. Tables and structures can also be considered and used as a component in a structure. A database table always has a structure therefore it is implicitly structure types. However, a field of the database tables represents elementary types.
- **Table types:** Table Types defines the complex types. *Table types* describe the structure and functions of the internal tables which is used in the ABAP program. Table types are considered as a construction blueprint for internal tables. When table type is created then line type, access type and key needs to be specified

All the semantic information for a type can be entered in the type definition in the ABAP Dictionary. This includes text that is displayed for the F1 help, search helps, text used in screens and technical documentation. Any complex types can be defined globally in the ABAP dictionary and can be used in ABAP programs. The runtime object of dictionary type (nametab) is the interface for their use in ABAP programs. The runtime object permits very efficient access to the relevant information for the type in compressed form.

When the type is changed, all the objects that use this type automatically adjusted to the change during activation. The central definition of types that are used more than once in ABAP Dictionary changes centrally. These changes are made at all the relevant location by the active ABAP Dictionary.

Note: All Dictionary types lie in a common namespace. A data element, structure and table type therefore, cannot have the same name. However, type defined in the dictionary and ABAP program may have the same name. If the names are identical then local types shadow the types of the type groups i.e. local types get the preference over global types or the types defined in the type groups.

Related Content

[BC-ABAP Dictionary](#)

[BC- ABAP Programming](#)

For more information, visit the [ABAP homepage](#).

Copyright

© Copyright 2010 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.