CAD Interface (CA-CAD)

HELP.CACAD

Release 4.6C

SAP™

Copyright

© Copyright 2001 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®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of
Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation.

INFORMIX®-OnLine for SAP and Informix® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

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

HTML, DHTML, XML, XHTML 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, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Inhalt

CAD Interface (CA-CAD) 14

Introduction to the CAD Interface ... 15
Before You Use the CAD Interface ... 18
This Guide is Intended for: ... 19
Use of the CAD Interface ... 20
Applications of the CAD Interface ... 22
Structure of the CAD Interface ... 23
CAD Interfaces in the SAP System ... 24
Dialog Interface ... 25
Dialog RFC Interface ... 27
Comparison of the Dialog Interface and the Dialog RFC Interface ... 29
Integration of the Dialog Interface and the Dialog RFC Interface ... 31
Installing Functions in the External Application ... 32
Overview of Functions in the CAD Dialog Interface ... 33
Material Master Record Functions (CAD Dialog Interface) ... 34
Document Management Functions (CAD Dialog Interface) ... 36
BOM Functions (CAD Dialog Interface) ... 37
Engineering Change Management Functions (CAD Dialog Interface) ... 39
Plant Maintenance Functions (CAD Dialog Interface) ... 40
General Functions (CAD Dialog Interface) ... 41
Overview of Functions in the Dialog RFC Interface ... 42
Material Master Record Functions (Dialog RFC Interface) ... 43
Document Management Functions (Dialog RFC Interface) ... 44
BOM Functions (Dialog RFC Interface) ... 45
Engineering Change Management Functions (Dialog RFC Interface) ... 46
Plant Maintenance Functions (Dialog RFC Interface) ... 47
Mail and Workflow Functions (Dialog RFC Interface) ... 48
General Functions (Dialog RFC Interface) ... 49
Examples of How the CAD Dialog Interface Can be Used ... 50
Finding a Document Info Record via Matchcode ... 51
Designing a Single-Part Drawing ... 53
Designing a Single-Part Drawing by Copying ... 55
Designing an Assembly Drawing ... 57
CAD Interface (CA-CAD) ... 59
Technical Description of the CAD Dialog Interface ... 60
General Information on the CAD Dialog Interface ... 61
SAP Function Library ... 62
SAP Interface Program ... 63
Notes on Installation ... 64
Customizing the SAP-CAD Function Library ... 65
Before You Use the SAP-CAD Function Library ... 66
Configuration File sideinfo ... 67
Configuration File saprfc.ini ... 69
Customizing the SAP System ... 70
General Information on Customizing ... 71
Control Parameters ... 72

Customizing General Data ... 74
User-Defined Functions ... 76
Example: Settings for Displaying a Where-Used List ... 77
Example: Settings for a User-Defined Function ... 79
Including RCCADCOM in User-Defined Functions ... 84
Emulation of the SAP System ... 85
Emulation Library ... 86
Data File cademu.r3 ... 87
General Functions in the CAD Dialog Interface ... 89
General Information ... 90
Data Transfer Using Data Strings ... 91
General Functions in the CAD Interface ... 93
CONNC - Establishing a Connection to the SAP System ... 94
DISCO - Terminating a Connection to the SAP System ... 95
BREAK - Terminating the Current Function ... 96
SYSRQ - System Functions ... 97
Displaying System Parameters ... 98
Displaying Configuration Tables TCIM, TCIU, and TCID ... 100
Displaying Technical Field Information for a Function ... 102
HLPRQ - Help Functions ... 103
Displaying ONLINE Field Documentation (F1 Help) ... 104
Displaying Possible Entries (F4 Help) ... 106
Displaying Allowed Values for Several Characteristics ... 108
Displaying the Long Text for an Error Message ... 109
Functions for Material Master Records ... 110
MATCR - Creating a Material Master Record ... 111
Creating a Material Master Record Without Classification ... 112
Creating a Material Master Record With Classification ... 113
Setting up Strings for Characteristics and Values ... 115
Characteristics Data for a Class ... 116
Values for Characteristics ... 119
Assigned Characteristic Values ... 120
MATCH - Changing a Material Master Record ... 122
Changing a Material Master Record Without Classification ... 123
Changing A Material Master Record With Classification ... 124
List of All Classes ... 126
MATCH MC: Creating an Allocation to an Additional Class ... 127
MATCH MC: Deleting an Existing Allocation to a Class ... 128
MATRQ - Displaying a Material Master Record ... 129
Displaying a Material Master Record Without Classification ... 130
Displaying a Material Master Record With Classification ... 131
Finding a Material Master Record via Matchcode ... 133
Displaying Material Master Record Data for Several Materials (Mass Access) ... 136
RSRVE - Reserving a Material Number ... 137
Functions for Document Management ... 138
DOCCR - Creating a Document Info Record ... 139

Creating a Document Info Record Without Classification 140
Creating a Document Info Record With Classification 141
DOCCH - Changing a Document Info Record 143
Changing a Document Info Record Without Classification 144
Changing a Document Info Record With Classification 145
Changing a Document Info Record (Object Links Only) 148
DOCRQ - Displaying a Document Info Record 151
Displaying a Document Info Record Without Classification 152
Displaying a Document Info Record With Classification 153
Finding a Document Info Record via Matchcode 155
Finding a Document Info Record Using Transaction CV04 158
Displaying a Document Info Record (Object Links Only) 161
Displaying the Status History of a Document Info Record 164
DOCCI - Check In Functions for Original Application Files 165
Check In for Original Application Files 166
Displaying a List of All Available Vaults 168
DOCCO - Check Out Functions for Original Application Files 169
Check Out for Original Application Files 170
Displaying an Original Application File 172
Functions for BOMs 174
BOMCR – Creating a BOM 175
BOMCN – Changing a BOM 177
BOMRQ – Displaying a BOM 179
BOMWU – Displaying a Where-Used List 181
CadBomCreateSubItem – Creating a BOM With Sub-Items 182
CadBomChangeSubItemRead – Reading and Changing a BOM 183
CadBomChangeSubItem – Changing a BOM with Sub-Items 184
CadBomDisplaySubItem – Displaying a BOM With Sub-Items 185
Functions for Engineering Change Management 186
CHGCR - Creating a Change Master Record 187
Creating a Change Master Record Without Classification 188
Creating a Change Master Record With Classification 190
CHGCH - Changing a Change Master Record 193
Changing a Change Master Record Without Classification 194
Changing a Change Master Record with Classification 196
CHGRQ - Displaying a Change Master Record 202
Displaying a Change Master Record Without Classification 203
Displaying a Change Master Record With Classification 205
Finding a Change Master Record via Matchcode 208
ECRCR - Creating an ECR 211
Creating an ECR Without Classification 212
Creating an ECR With Classification 214
ECRCH - Changing an ECR 217
Changing an ECR Without Classification 218
Changing an ECR with Classification 220

ECRRQ - Displaying an ECR ... 227
Displaying an ECR Without Classification ... 228
Displaying an ECR With Classification ... 230
Finding an ECR via Matchcode ... 233
Functions for Plant Maintenance ... 236
EQICR - Creating an Equipment Master Record ... 237
Creating an Equipment Master Record Without Classification ... 238
Creating an Equipment Master Record With Classification ... 239
EQICH - Changing an Equipment Master Record ... 241
Changing an Equipment Master Record Without Classification ... 242
Changing an Equipment Master Record With Classification ... 243
Dismantling Equipment Master Records ... 246
EQIRQ - Displaying an Equipment Master Record ... 247
Displaying an Equipment Master Record Without Classification ... 248
Displaying an Equipment Master Record With Classification ... 249
Finding an Equipment via Matchcode ... 251
Displaying an Equipment Hierarchy ... 254
PMNCR - Creating a Maintenance Notification ... 255
TPCCR - Creating a Functional Location ... 256
Creating a Functional Location Without Classification ... 257
Creating a Functional Location With Classification ... 258
TPCCH - Changing a Functional Location ... 260
Changing a Functional Location Without Classification ... 261
Changing a Functional Location With Classification ... 262
TPCRQ - Displaying a Functional Location ... 265
Displaying a Functional Location Without Classification ... 266
Displaying a Functional Location With Classification ... 267
Finding a Functional Location via Matchcode ... 269
Displaying a Functional Location Hierarchy ... 272
Displaying a Functional Location with Installed Equipments ... 273
Classification System Functions ... 274
SLCLS - Finding Classes - Navigating in a Class Network ... 275
Finding Classes by Catchwords (Matchcode) ... 276
Finding Classes by Generic Entry ... 279
Finding Classes without a Superior Class ... 281
Displaying Superior and Subordinate Classes for a Class ... 283
SLOBJ - Finding Objects in Classes ... 285
Displaying the Characteristics of a Class ... 286
Displaying All Objects in a Class ... 287
Finding Objects via Classes ... 290
Setting a New View in Table INDX ... 294
Initializing Class Data ... 295
Office Functions ... 296
SMAIL - Office Functions ... 297
Reading Your Inbox ... 298

Reading Your Outbox ... 299
Reading Your Mail Trash ... 300
Reading a Document ... 301
Reading the Attachments to a Document ... 303
Setting a Document to ... 304
Setting a Document to ... 305
Deleting a Document ... 306
Displaying the Recipient List for a Document ... 307
Sending a Document ... 308
Finding an SAP User Name Generically ... 309
Description of Function Library caddialg ... 310
General Information on the Function Library ... 311
Header Files and Libraries Required ... 312
Structure of Function Calls ... 313
Transfer Fields ... 314
Status Information ... 315
Syntax ... 317
Description of Functions ... 318
General functions ... 319
Functions for Material Master Records ... 322
Classification System Functions ... 323
Functions for Document Management ... 324
Functions for Plant Maintenance ... 325
Functions for Engineering Change Management ... 326
Functions for BOMs ... 327
Office Functions ... 329
User-Defined Functions ... 330
General Rules for Data Transfer ... 331
Functions of the CAD Toolbox Library ... 332
Introduction ... 333
Installation and Implementation ... 334
Functions ... 335
General Functions ... 336
Configure Toolbox ... 337
Init Toolbox ... 338
Load Configuration Data ... 339
Load Error Message ... 340
Load Version of Toolbox ... 341
Save Configuration Parameters ... 342
Save Configuration Tables ... 343
Utility Functions ... 344
Delete Table Entry ... 345
Load Attribute Name ... 346
Print Internal Table ... 347
Find Attribute ... 348

Master Record Functions ... 349
Count Table Entries (Master Record Functions) ... 350
Create Internal Table ... 351
Create and Fill Internal Table ... 352
Create Send String ... 354
Create Send String for BOM Header Data (R/3) ... 355
Fill Internal Table ... 356
Load Attribute Value for Record Number ... 357
Load Attribute Value for Attribute Name ... 358
Change Table Value ... 360
Save Table Value ... 361
Classification Functions ... 362
Count Table Entries (Classification Functions) ... 363
Count Characteristic Values (R/3) ... 364
Create Send String for Allowed Characteristic Values (R/3) ... 365
Create Send String for Characteristics Data (R/2 and R/3) ... 366
Create Send String for Class Data (R/3) ... 367
Create Send String for Class Scope (R/3) ... 368
Create Send String for Finding Classes(R/3) ... 369
Create Send String for Finding Objects via Classes (R/3) ... 370
Create and Fill Class Hierarchy Table(R/3) ... 371
Create and Fill Classification Table (R/3) ... 372
Create and Fill Characteristic Values Table(R/3) ... 373
Create Characteristics Data Table ... 374
Delete Search Result (R/2) ... 375
Load Classification Data for Characteristic Number (R/2) ... 376
Load Classification Data for Characteristic Name (R/2) ... 377
Load Classification Data for Characteristic Number (R/3) ... 378
Load Classification Data for Characteristic Name (R/3) ... 379
Load Characteristic Value Data (R/3) ... 380
Load Object Oata (R/3) ... 381
Load Class Data (R/3) ... 382
Load Class Hierarchy Data (R/3) ... 383
Load Search Result Data (R/3) ... 384
Save Characteristic Value and Description ... 385
Save Search Result ... 388
Matchcode Functions ... 389
Count Matchcode Records ... 390
Create and Fill Internal Matchcode Table ... 391
Create Send String for Matchcode ... 392
Load Matchcode Fields ... 393
Save Matchcode Field Value ... 394
Online Help Functions ... 395
Count Table Entries (Help Functions) ... 396
Fill Internal Table for Online Help ... 397

Create Send String for Online Help ... 398
Create Send String for Long Text of SAP Message (R/3) ... 399
Fill Internal Table for Online Help ... 400
Load Online Help Data ... 401
Mail Functions (R/3 2.1/2.2) ... 402
Count Table Entries (Mail Functions) ... 403
Create Send String for Mail Data ... 404
Create Internal Table for Mail Data ... 405
Load Mail Data from Table ... 406
Mail Field Names ... 407
Save Mail Data to Table ... 409
User-Defined Functions ... 410
Create Internal Table for User-Defined Functions ... 411
Appendix ... 412
Files ... 413
CADTMES.DAT ... 414
CADTTAB.DAT ... 418
CADTOOLC.H ... 420
Structures ... 428
SAPOBJ_R3 ... 429
SAPSML_R3 ... 430
SAPCLAS_R3 ... 432
SAPCLAS_S_R3 ... 433
SAPCLAS_H_R3 ... 434
SAPMES ... 435
SAPATT ... 436
Tables ... 437
TCIM ... 438
TCIT ... 440
TCID ... 444
TCIU ... 447
Technical Description of the Dialog RFC Interface ... 454
General Information on the Dialog RFC Interface ... 455
How the Dialog RFC Interface Works ... 456
Applications of the Dialog RFC Interface ... 457
Notes on Installation ... 458
Customizing the Dialog RFC Interface ... 459
Configuration File cadrfc.ini ... 460
Default Values for BOMs ... 464
Default Values for Document Management ... 467
Default Values for Material Master Records ... 468
Default Values for Plant Maintenance ... 469
General Default Values ... 470
Description of the Available RFC Function Modules ... 471
Transfer Tables SAP_FIELD_DATA and DMS_CLASS_DATA ... 472
Functions for Document Management ... 473

Creating a Document Info Record ... 474
Changing a Document Info Record ... 476
Displaying a Document Info Record ... 478
Finding a Document Info Record ... 480
Inserting a Document Info Record ... 482
Reading a Document Info Record ... 484
Inserting and Checking Out Documents ... 486
Reading Data for the Drawing Header ... 487
Functions for Material Master Records ... 489
Creating a Material Master Record ... 490
Changing a Material Master Record ... 492
Displaying a Material Master Record ... 494
Reading a Material Master Record ... 496
Functions for Engineering Change Management ... 498
Creating a Change Master Record ... 499
Changing a Change Master Record ... 501
Displaying a Change Master Record ... 503
Functions for BOMs ... 504
Creating a Material BOM ... 505
Changing a Material BOM ... 507
Displaying a Material BOM ... 509
Functions for Plant Maintenance ... 512
Creating an Equipment Master Record ... 513
Changing an Equipment Master Record ... 515
Displaying an Equipment Master Record ... 516
Reading an Equipment Master Record ... 518
Creating a Functional Location ... 520
Changing a Functional Location ... 522
Displaying a Functional Location ... 524
Reading a Functional Location ... 526
Creating a Maintenance Notification ... 528
Changing a Maintenance Notification ... 530
Displaying a Maintenance Notification ... 531
Creating a Maintenance Order ... 532
Changing a Maintenance Order ... 534
Displaying a Maintenance Order ... 535
Mail and Workflow Functions ... 536
Displaying a Mail Inbox ... 537
Displaying a Mail Outbox ... 538
Displaying a Workflow Inbox ... 539
General Functions in the SAP Environment ... 540
Displaying a Product Structure ... 541
Displaying a Where-Used List for a Material or Document ... 543
Finding Objects via Matchcode ... 545
Finding Objects in Classes ... 547
Requesting SAP System Parameters ... 550

Description of Dialog RFC Function Library cadrfc00 551
The Dialog RFC Function Library..... 553
Header Files and Libraries Required 554
Header File cadrfc.h..... 555
Table Structures..... 557
General Functions..... 558
Establishing an RFC Connection (Starting from External System) 559
Terminating an RFC Connection 561
Initializing Transfer Structures 562
Querying a Communication Handle..... 563
Defining a Communication Handle 564
Return of Error Messages..... 565
Functions for Material Master Records 566
Creating a Material Master Record..... 567
Changing a Material Master Record 568
Displaying a Material Master Record..... 569
Reading a Material Master Record 570
Functions for Document Management..... 571
Creating a Document Info Record 572
Changing a Document Info Record 573
Displaying a Document Info Record 574
Finding a Document Info Record 575
Inserting a Document Info Record 576
Reading a Document Info Record 577
Functions for BOMs 578
Creating a Material BOM 579
Changing a Material BOM..... 580
Displaying a Material BOM 581
Filling out the BOM Item Table 582
Reading the BOM Item Table 583
Functions for Engineering Change Management 584
Creating a Change Master Record 585
Changing a Change Master Record 586
Displaying a Change Master Record 587
Functions for Plant Maintenance..... 588
Creating an Equipment Master Record 589
Changing an Equipment Master Record..... 590
Displaying an Equipment Master Record 591
Reading an Equipment Master Record..... 592
Creating a Functional Location 593
Changing a Functional Location 594
Displaying a Functional Location 595
Reading a Functional Location 596
Creating a Maintenance Notification..... 597
Changing a Maintenance Notification 598
Displaying a Maintenance Notification..... 599

Creating a Maintenance Order ... 600
Changing a Maintenance Order ... 601
Displaying a Maintenance Order ... 602
Mail and Workflow Functions ... 603
Displaying a Mail Inbox ... 604
Displaying a Mail Outbox ... 605
Displaying a Workflow Inbox ... 606
General Functions in the SAP Environment ... 607
Displaying a Product Structure ... 608
Requesting SAP System Parameters ... 609
Displaying a Where-Used List for a Material or Document ... 610
Finding Objects via Matchcode ... 611
Return of Last RFC Error Message ... 612
Reading an Entry in Table SAP_FIELD_DATA ... 613
Reading an Entry in Table DMS_CLASS_DATA ... 614
Making an Entry in Table SAP_FIELD_DATA ... 615
Making an Entry Table DMS_CLASS_DATA ... 616
Finding Objects in Classes ... 617
Example of a Call ... 619
Using Calls in the SAP-CAD Function Library ... 620
Using the CAD Desktop ... 621
Integrating Enterprise-Specific Processes ... 624
Appendix ... 625
Error messages in function library caddialg ... 626
CPIC Communication Errors and CPIC Error Codes ... 628
Return Codes from the SAP System ... 630
Return codes of the SAP-CAD function library ... 631
Return Codes in the cadrfc00 Library ... 632
File sideinfo ... 635
File saprfc.ini ... 636
File cadrfc.ini ... 643
File caddialg.h ... 653
File cadrfc.h ... 656
File cadtst1.c ... 670

CAD Interface (CA-CAD)

Purpose

CAD integration is connecting the SAP System to external applications such as CAD, PDM, and GIS.

This enables your designers to access data in the production system and use this information in their work - information on existing materials and assemblies, for example. Since data can be transferred between the CAD system and the SAP System in both directions, inconsistency of the data in the two systems is avoided.

Implementation considerations

These components are necessary when you want to couple or link the SAP System to an external application.

See also: Introduction to the CAD Interface [Seite 15]

Integration

You can use the CAD interface for the following applications to transfer data online between an external system and your SAP System:

- General Functions
- Functions for Material Master Records
- Functions for Document Management
- Functions for BOMs
- Functions for Plant Maintenance
- Functions for Engineering Change Management
- Mail and Workflow Functions

Features

Depending on the area of usage you can select between the CAD dialog interface and the dialog RFC interface. The following is an overview:

CAD dialog interface	Dialog RFC Interface
Overview of Functions in the CAD Dialog Interface [Seite 33]	Overview of Functions in the Dialog RFC Interface [Seite 42]
Technical Description of the CAD Dialog Interface [Seite 60]	Technical Description of the Dialog RFC Interface [Seite 454]

Introduction to the CAD Interface

This section contains information on the content and structure of the documentation on the CAD interface. It also describes the tasks and the possible uses of the CAD interface.

Introduction to the CAD Interface

Before You Use the CAD Interface [Seite 18]

This Guide is Intended for: [Seite 19]

Use of the CAD Interface [Seite 20]

Applications of the CAD Interface [Seite 22]

Structure of the CAD Interface [Seite 23]

CAD Interfaces in the SAP System [Seite 24]

Comparison of the Dialog Interface and the Dialog RFC Interface [Seite 29]

Integration of the Dialog Interface and the Dialog RFC Interface [Seite 31]

Installing Functions in the External Application [Seite 32]

Overview of Functions in the CAD Dialog Interface [Seite 33]

General Functions (CAD Dialog Interface) [Seite 41]

Material Master Record Functions (CAD Dialog Interface) [Seite 34]

Document Management Functions (CAD Dialog Interface) [Seite 36]

BOM Functions (CAD Dialog Interface) [Seite 37]

Plant Maintenance Functions (CAD Dialog Interface) [Seite 40]

Engineering Change Management Functions (CAD Dialog Interface) [Seite 39]

Overview of Functions in the Dialog RFC Interface [Seite 42]

General Functions (Dialog RFC Interface) [Seite 49]

Document Management Functions (Dialog RFC Interface) [Seite 44]

Material Master Record Functions (Dialog RFC Interface) [Seite 43]

Engineering Change Management Functions (Dialog RFC Interface) [Seite 46]

Mail and Workflow Functions (Dialog RFC Interface) [Seite 48]

BOM Functions (Dialog RFC Interface) [Seite 45]

Plant Maintenance Functions (Dialog RFC Interface) [Seite 47]

Examples of How the CAD Dialog Interface can be Used [Seite 50]

Finding a Document Info Record via Matchcode [Seite 51]

Designing a Single-Part Drawing [Seite 53]

Designing a Single-Part Drawing by Copying [Seite 55]

Designing an Assembly Drawing [Seite 57]

Before You Use the CAD Interface

The following sections are an introduction to the CAD interface:

- This Guide is Intended for: [Seite 19]
- Use of the CAD Interface [Seite 20]
- Applications of the CAD Interface [Seite 22]

This Guide is Intended for:

This guide is mainly intended for the following three target groups:

- Engineers, designers, and technicians who work with a CAD, PDM, or GIS system and process data in a connected R/3 System via the CAD interface.

  For this target group, the guide provides an overview of the range of functions in the interface and the essential prerequirements for its use.
- Application developers who are using the CAD interface to write an interface program between a CAD, PDM, or ITS system and the SAP System.
- System administrators who configure and maintain the CAD interface.

Use of the CAD Interface

Information systems in a company tend to be divided into two main areas:

- Engineering information systems
- Commercial information systems

Most companies have tended to develop these two areas independently from each other, the main concern being how well the system could solve the specific problems of the area concerned. Integration with other applications was only a minor consideration. The isolated solutions arising from this situation lead to a number of problems:

- Data needed in more than one system has to be entered manually several times and subsequently updated manually several times.
- As it is not possible to access data from other application areas, product development work may be duplicated or parallel developments may result. You do not have a central overview of existing data.
- Central control and monitoring of projects and transactions involving several departments is only possible with extensive organization.

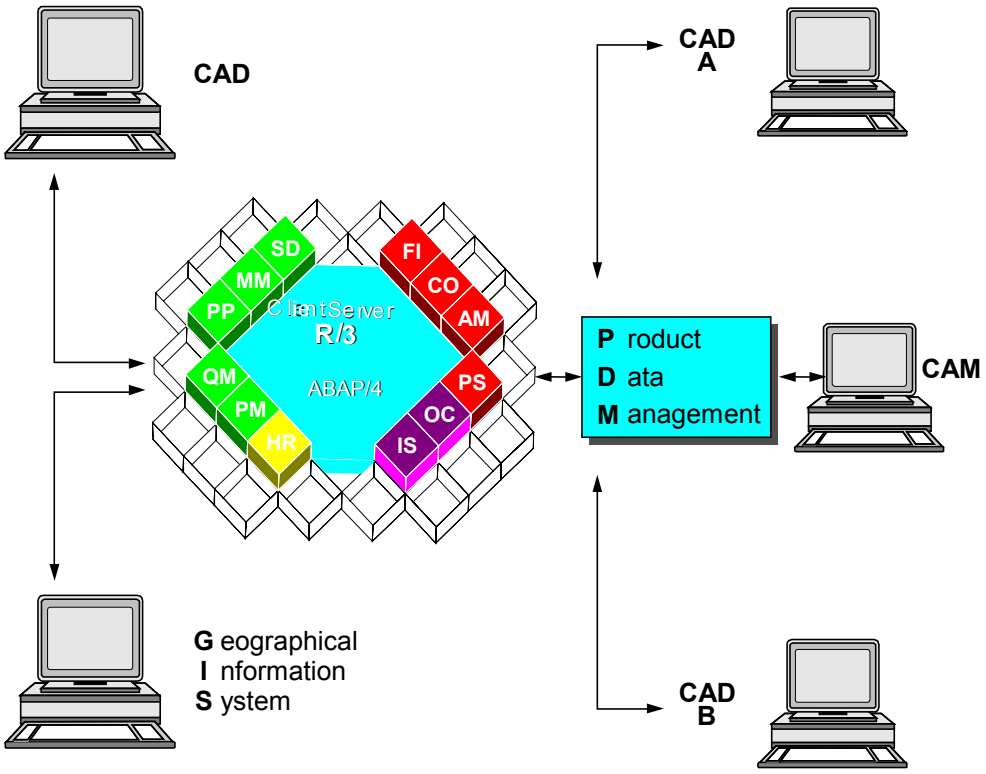
Since flexibility and efficiency are becoming increasingly important to the performance of a company, the integration of information systems in different departments is becoming one of a company's most important priorities.

The essential aims of this process are:

- Central storage of data to minimize data redundancy and reduce the data maintenance workload
- Connections between and networking of the individual information systems, particularly between engineering and commercial application areas
- Quick access to existing data
- Central control, analysis, and monitoring of processes that involve more than one area

If your company uses an SAP System for processing commercial data, this system can be connected to CAD work centers in your engineering information system using the CAD dialog interface. This enables your designers to access data in the production system and use this information in their work - information on existing materials and assemblies, for example. Since data can be transferred between the CAD system and the SAP System in both directions, inconsistency of the data in the two systems is avoided.

Data Transfer Between an External System and the SAP System



Applications of the CAD Interface

You can use the CAD interface for the following applications to transfer data online between an external system and your SAP System:

- Document management
- Material master records
- Bills of material
- Engineering change management
- SAP mail and workflow
- Plant maintenance
- Classification
- Configuration

Structure of the CAD Interface

This section contains information on the two CAD interfaces in the SAP System. The main distinguishing features of the two interfaces are compared and you see how the functions of the CAD interface can be used by an external application.

- CAD Interfaces in the SAP System [Seite 24]
- Dialog Interface [Seite 25]
- Dialog RFC Interface [Seite 27]
- Comparison of the Dialog Interface and the Dialog RFC Interface [Seite 29]
- Integration of the Dialog Interface and the Dialog RFC Interface [Seite 31]
- Installing Functions in the External Application [Seite 32]

CAD Interfaces in the SAP System

You can choose between the following options, according to where you want to use your interface:

- Dialog Interface [Seite 25]
- Dialog RFC Interface [Seite 27]

Dialog Interface

The dialog interface has been available since R/3 2.1 and was enhanced considerably for release 3.0.

From your CAD system, you can transfer data online both from and to your R/3 System. The data you request is read from the SAP database and converted to the format required for transfer. Data that is transferred from the external system is subjected to the same consistency checks that are made online before the SAP database is updated.

The SAP System functions as a server. Any dialog required is carried out via specially defined windows in the external application. In a LAN/WAN, the interface uses the TCP/IP protocol.

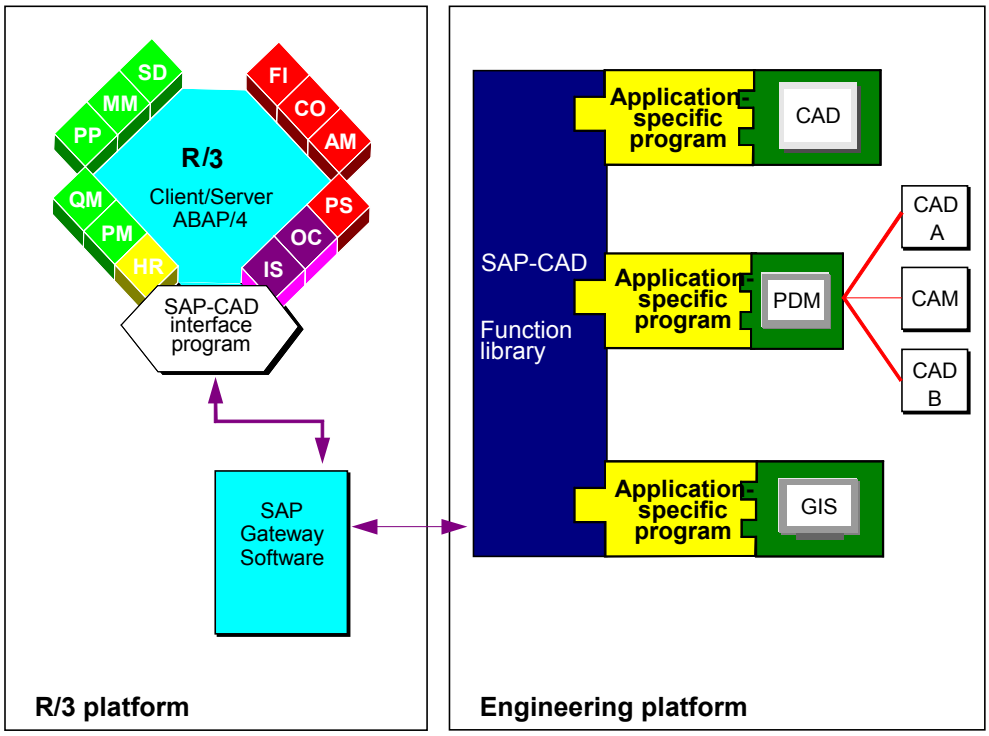
The interface comprises the following two components:

- SAP-CAD function library on the engineering platform
- SAP interface program, integrated in the R/3 System

Before you can access the individual API functions from your external system, the SAP-CAD function library must be connected to the engineering application using an application-specific interface program.

The figure below shows the transfer of data between the two platforms.

R/3 Platform and Engineering Platform



Dialog RFC Interface

When you work with the dialog RFC interface (**R**emote **F**unction **C**all), an RFC connection is established between the active SAP-GUI window and an external application.

Once this connection is established, function calls can be carried out from the external system as required. Since the connection is via an active GUI session, these function calls can also contain SAP dialog screens (dialog RFC).

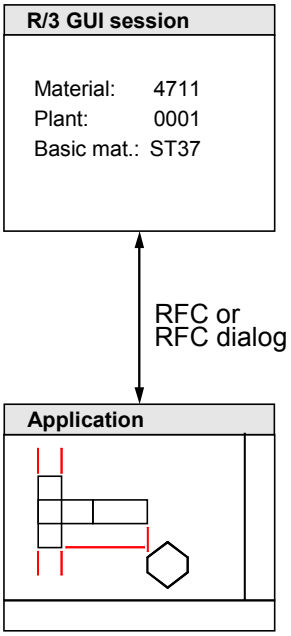
In releases previous to 3.0C, the connection can only be started from the R/3 System under normal circumstances. As of release 3.0C, you can also start the connection from your external system.

A whole range of function modules that can be called by *Remote Function Call* has been developed especially for the dialog RFC interface. Most of these modules allow you to access SAP dialog transactions easily, and to transfer data.

If required, you can also use all the API function calls of the dialog interface when working with the dialog RFC interface.

The figure below illustrates the general workings of the RFC dialog interface.

Connection Between the R/3 System and an External Application



Comparison of the Dialog Interface and the Dialog RFC Interface

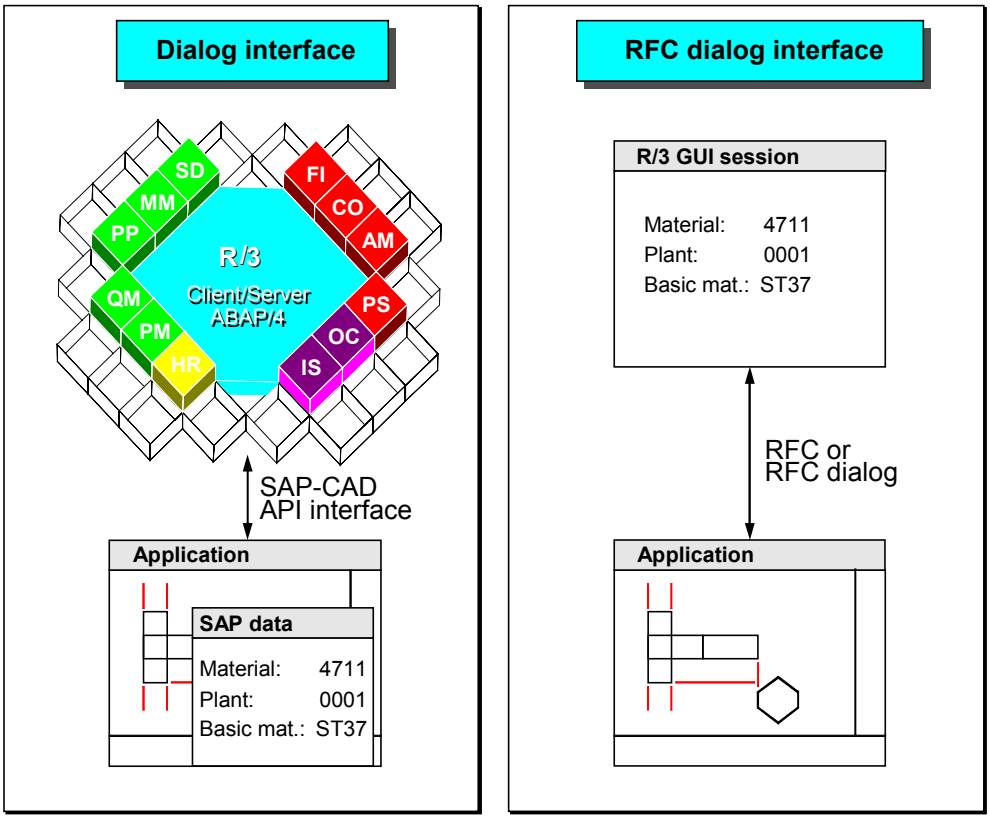
The following table compares the distinguishing features of these two interfaces.

Comparison of the Dialog Interface and the Dialog RFC Interface

	Dialog Interface	**Dialog RFC Interface**
Dialog	Is emulated in the external application	SAP screens can be used
Active system	R/3 System functions as database server; External application is the leading system	R/3 System and external application are active in alternation
Communication	Via API calls in the SAP-CAD function library	Via Remote Function Calls (RFC)
Data transfer	Via defined data strings	Via RFC data structures (fields, structures, tables)
Interface enhancements	User-defined APIs	Integration of any dialog RFCs or RFCs

The figure below illustrates the differences between the two interfaces.

Comparison of the Dialog Interface and the Dialog RFC Interface



Integration of the Dialog Interface and the Dialog RFC Interface

When you work with the dialog RFC interface, you can also use the whole range of API calls in the dialog interface. This allows you to transfer data to and from the R/3 System in the background.

For example, if you are doing drawing administration you can use regular SAP screens, but you can transfer a BOM without SAP dialog using the APIs in the dialog interface.

Installing Functions in the External Application

Whether and to what extent the range of functions in the CAD interface is available to users of an external system depends on which functions and parameters from the CAD interface the external system interface program uses.

The functions can be made accessed by external system users in different ways:

- by entering a command string in a field defined for this purpose
- by selecting a function from the menu

For details of how the CAD interface is implemented in your engineering program, please refer to the documentation for the interface program concerned.

Overview of Functions in the CAD Dialog Interface

This section describes all the functions the CAD dialog interface. You can use functions of the CAD dialog interface for the following application areas:

- General Functions (CAD Dialog Interface) [Seite 41]
- Material Master Record Functions (CAD Dialog Interface) [Seite 34]
- Document Management Functions (CAD Dialog Interface) [Seite 36]
- BOM Functions (CAD Dialog Interface) [Seite 37]
- Plant Maintenance Functions (CAD Dialog Interface) [Seite 40]
- Engineering Change Management Functions (CAD Dialog Interface) [Seite 39]

See also:

Functions in the CAD Dialog Interface (I) [Seite 89]

Material Master Record Functions (CAD Dialog Interface)

The following functions for accessing material master data are supported in the SAP-CAD dialog interface:

- **Creating a Material Master Record**

  You can use the SAP-CAD dislog interface to create material master records in the SAP System from your external system. You can create all views on a material.

  When you create a material, you can either enter the *material number* or leave it to be assigned from an internal number range. You need to define the *material type* and the *industry sector*.

- **Reserving a material number**

  When you create a new material in the external system the material is saved to the database of the external system under a specific number. When the drawing is released, a material master record for this material must be created in the SAP System. To ensure that the number you require is still available when you come to create the material in the SAP System, you have the option of reserving a number from an internal number range in the SAP System from your external system. The number is transferred to the external system and can then be displayed and assigned to the material.

  The SAP System counts up the next number in the internal number range, but no material master record is created. Once the design phase of the new material is completed, you can create the material in the SAP System under the number you reserved.

  This process ensures that you do not have to rename a material you created in the external system under a specific number when you create the material in the SAP System using the function described above.

- **Changing a Material Master Record**

  You can change all views on a material from your external system. Settings in the relevant tables determine which data is transferred from the SAP System to the external system and back. You have to enter the of the number of the material you want to change. Depending on which fields from the SAP material master you want to change, you may have to make entries in other fields (such as *Plant* or *Storage location*).

  If the material you want to change is not found in the SAP System, the function is terminated. If a material that has the number you entered is found, the data of this material is transferred to the external system. At the same time, the material is locked for changes by other users in the SAP System.

- **Displaying a Material Master Record**

  You can use this function to display the data of a material from the SAP System. The only prerequisite for this is that the user name you used to log on to the SAP System with is authorized to display material master data.

  The following display functions are supported:

  – *Display material master record without classification data*

You need to enter the number of the material you want to display. Depending on which fields from the SAP material master you want to change, you may have to make entries in other fields (such as *Plant* or *Storage location*).

– *Find material via matchcode*

After the request to find a material via matchcode has been transferred to the SAP System, the SAP System sends all the available matchcodes to the CAD System. If you know the structure of the matchcode that you want to use, enter the required data next to the matchcode ID. The SAP System determines which material numbers match this data and transfers them to the external system. After you have chosen one of these materials, its data is displayed. If you do not know the structure of the matchcode that you want to use, just select the matchcode ID. The SAP System transfers the fields of this matchcode to the external system. Once you have made your entries, you see a list of materials that match your selection criteria.

Document Management Functions (CAD Dialog Interface)

The SAP-CAD dialog interface can be used to transfer data in document info records between the SAP System and the external system.

The following functions are supported:

- **Creating a Document Info Record**

  When you create a document info record from your external system, you can either enter a document number or have the number assigned by the SAP System. If you want the number to be assigned by the SAP System, the special character '*' or a blank must be transferred from the external system to the SAP System via the interface. The number determined by the SAP System is transferred to the external system and can then be assigned to the document for which you want to create the document info record. Version assignment for the documents can also be internal or external. The *document type* and *document part* are required entries.

- **Changing a Document Info Record**

  This function allows you to change document info records stored in the SAP System from your external system.

  To select a document info record, the document number, document type, document part, and document version must be transferred to the SAP System. If the document info record does not exist in the SAP System, the function is terminated. Otherwise, the data in the document info record is transferred to the external system and can be changed there. While you are changing the document info record in the external system, the record is locked in the SAP System.

- **Displaying a Document Info Record**

  You can use this function to display the data of a document info record from the SAP System. The only prerequisite for this is that the user name you used to log on to the SAP System with is authorized to display document info records of the chosen document type.

  The following display functions are supported:

  – *Display document info record with/without classification data*

    To identify the document info record you want to display, the document number, document type, document part, and version number must be transferred to the SAP System.

Find document info record via matchcode

  – To display a document using a matchcode, first select the matchcode ID. If you know the structure of the matchcode you want to use, enter the matchcode ID followed by the data required. Otherwise, select the matchcode ID only. The SAP System transfers the fields of this matchcode to the external system. Once you have made your entries, the document info records that match your selection criteria are listed.

BOM Functions (CAD Dialog Interface)

The SAP:CAD dialog interface can be used to transfer BOM data between the SAP System and your external system.

The following functions are supported:

- **Create BOM**

  A BOM in the SAP System comprises two main parts: the *BOM header* and the *BOM items*. Both parts must be entered and transferred to the SAP System when you create a BOM in the external system.

  You cannot make entries in the following fields:

  – *Long text indicator*

    You can only enter long texts from the SAP System.

  – *CAD indicator*

    This indicator is set automatically by the SAP interface program.

    You can only create multiple BOMs, and not variant BOMs.

As of release 4.0A, you can also create BOMs with sub-items. You can create sub-items for each BOM item. They are often used to determine the installation point of an item more exactly.

- **Change BOM**

  This function allows you to change BOMs stored in the SAP System from your external system.

  To uniquely identify the BOM, you must enter the *material number* of the BOM header material, the *plant*, and the *usage*. If the BOM is to be maintained with history, you also need to enter a *change number*. Otherwise, changes can also be made without entering a change number.

  You can only change a field in a BOM from the external system if it is one of the fields in which you can make an entry when creating a BOM from the external system. Only BOM items that were created from the external system, and therefore contain the CAD indicator, can be changed from the external system.

  If the BOM or change number you enter does not exist in the SAP System, or is not valid on the date specified, the SAP interface program terminates the function. Otherwise, the BOM header data and all CAD items are transferred to the external system. Once you have made your changes, all of this data is transferred back to the SAP System via the dialog interface. The SAP System compares the data it receives with the data previously stored in the system, and determines which items are to be updated in the database.

  As of release 4.0A, you can also change BOMs with sub-items using the CAD dialog interface.

- **Display BOM**

BOM Functions (CAD Dialog Interface)

You can use this function to display BOM data from the SAP System. The user you logged on with must have the appropriate authorization for displaying BOMs of this BOM category and plant. The following display functions are supported:

- *Display BOM (single-level)*
- *Display BOM level by level*
- *Display multi-level BOM*
- *Display summarized BOM*

To identify the BOM you want to display completely, you must enter the material, the plant, and the usage. For multiple BOMs, you must enter the alternative or the required quantity. If no valid from date is defined in the CAD System, the SAP System uses the current date.

As of release 4.0A, you can display BOMs with sub-items using the CAD dialog interface.

Engineering Change Management Functions (CAD Dialog Interface)

The CAD interface allows you to transfer data from the *Engineering Change Management* system between the SAP System and an external system.

The following functions are supported:

- **Creating a Change Master Record**

  The CAD interface allows you to create change master records in the SAP System from the external application program. You can create all data in the change master record with the exception of long texts. When you create a new change master record, you can classify the change master record. You can also set the indicators that define how the change master record can be used in changes.

- **Changing a Change Master Record**

  You can change the data in a change master record with and without classification data.

- **Displaying a Change Master Record**

  You can display functional location data in a change master record with and without classification data.

  The CAD interface gives you the option of finding a change master record if you do not know the change number, but do know some other master data, such as the description or valid-from date. You can also use the find objects function in the classification system to find change master records.

- **Create ECR**

  The CAD interface allows you to create ECRs in the SAP System from your external application program. You can create all data in the ECR with the exception of long texts. When you create a new ECR, you can maintain its classification data and the indicators that control its use in changes.

- **Change ECR**

  You can change the data in an ECR with and without classification data.

- **Display ECR**

  You can display the data in an ECR with and without classification data.

  The CAD interface gives you the option of finding an ECR if you do not know the change number, but do know some other master data, such as the description or valid-from date. You can also use the find objects function in the classification system to find ECRs.

Plant Maintenance Functions (CAD Dialog Interface)

The CAD interface allows you to transfer data from plant maintenance objects between the SAP System and an external system.

The following functions are supported:

- **Creating an Equipment Master Record**

  The CAD interface allows you to create equipment master records in the SAP System from the external application program. You can create all data in the equipment master record, with the exception of long texts. When you create a new equipment master record, you can install this equipment at a functional location immediately.

- **Changing an Equipment Master Record**

  You can change the data in an equipment master record with and without classification data.

- **Displaying an Equipment Master Record**

  You can display the data in an equipment master record with and without classification data.

  The CAD interface gives you the option of finding an equipment if you do not know the equipment number, but do know some other master data, such as the description or cost center. You can also use the find objects function in the classification system to find equipments. This function is described in the section on the classification system.

  You can also display the hierarchy that contains the current equipment.

- **Creating a Maintenance Notification**

  The maintenance notification is created in the SAP System, and a message is sent to the external application program.

- **Creating a Functional Location**

  You can create functional locations in the SAP System from your external application program. You can maintain all functional location data, with the exception of long texts.

- **Changing a Functional Location**

  You can change the data in a functional location with and without classification data.

- **Displaying a Functional Location**

  You can display functional location data in a change master record with and without classification data.

  The CAD interface gives you the option of finding a functional location if you do not know the functional location number, but do know some other master data, such as the description or cost center. You can also use classification system functions to find functional locations.

  You can also display the hierarchy that contains the current functional location.

General Functions (CAD Dialog Interface)

The following general functions are supported by the SAP-CAD dialog interface:

- **Logon to SAP System**

  You use this function to establish an online connection between the external system and your SAP System. If the connection is established successfully, the SAP System sends a message to the external system. This message contains the identifier of the SAP System and the release number.

- **Logoff from SAP System**

  You use this function to terminate an online connection between the external system and your SAP System. You can call this function at any time, even while you are processing another function. If you are processing another function, the function is terminated correctly before the connection is terminated.

- **Terminate function**

  You use this command to terminate a function that consists of several communication steps. You can terminate a function whenever the external system is doing the processing. The current function is interrupted, all temporary data is deleted, and the external system is then ready for you to enter a new command.

- **Synchronization**

  The format of the data transferred between the external system and the SAP System when they communicate with each other is defined in tables in the SAP System. These tables must be transferred to the external system, so that data sent is in the same format as data received. They must only be transferred if one of the tables has been changed in the SAP System. This function usually runs without the end user having to do anything.

- **Display online documentation**

  This function allows you to display the online documentation for a field, provided that documentation for the field exists in the SAP System.

- **Display possible entries**

  You can use this function to display the possible entry values for a field, provided that a check table has been allocated to this field in the SAP System.

Overview of Functions in the Dialog RFC Interface

You can use the dialog RFC interface to access all dialog RFCs and RFCs in the SAP System. A fixed component of the dialog RFC interface, designed especially for the engineering environment, is a range of standardized dialog RFC calls and RFC calls.

You can use standardized dialog RFC calls and RFC calls for the following functions:

- General Functions (Dialog RFC Interface) [Seite 49]
- Document Management Functions (Dialog RFC Interface) [Seite 44]
- Material Master Record Functions (Dialog RFC Interface) [Seite 43]
- Engineering Change Management Functions (Dialog RFC Interface) [Seite 46]
- Mail and Workflow Functions (Dialog RFC Interface) [Seite 48]
- BOM Functions (Dialog RFC Interface) [Seite 45]
- Plant Maintenance Functions (Dialog RFC Interface) [Seite 47]

Material Master Record Functions (Dialog RFC Interface)

You can use standardized dialog RFC calls and RFC calls for the following material master record functions:

- **Creating a Material Master Record**
- **Changing a Material Master Record**
- **Displaying a Material Master Record**

See also:

Functions for Material Master Records [Seite 489]

Document Management Functions (Dialog RFC Interface)

You can use standardized dialog RFC calls and RFC calls for the following document management functions:

- **Creating a Document Info Record**
- **Changing a Document Info Record**
- **Displaying a Document Info Record**
- **Finding a Document Info Record**
- **Inserting a Document Info Record**
- **Reading a Document Info Record**

See also:

Functions for Document Management [Seite 473]

BOM Functions (Dialog RFC Interface)

You can use standardized dialog RFC calls and RFC calls for the following BOM functions:

- **Creating a Material BOM**
- **Changing a Material BOM**
- **Displaying a Material BOM**

See also:

BOM Functions [Seite 504]

Engineering Change Management Functions (Dialog RFC Interface)

You can use standardized dialog RFC calls and RFC calls for the following *Engineering Change Management* functions:

- **Creating a Change Master Record**
- **Changing a Change Master Record**
- **Displaying a Change Master Record**

See also:

Functions for Engineering Change Management [Seite 498]

Plant Maintenance Functions (Dialog RFC Interface)

You can use standardized dialog RFC calls and RFC calls for the following plant maintenance functions:

- **Creating an Equipment Master Record**
- **Changing an Equipment Master Record**
- **Displaying an Equipment Master Record**
- **Creating a Functional Location**
- **Changing a Functional Location**
- **Displaying a Functional Location**
- **Creating a Maintenance Notification**
- **Changing a Maintenance Notification**
- **Displaying a Maintenance Notification**
- **Create maintenance order** (not yet supported in 3.0A)
- **Change maintenance order** (not yet supported in 3.0A)
- **Display maintenance order** (not yet supported in 3.0A)

See also:

Functions for Plant Maintenance [Seite 512]

Mail and Workflow Functions (Dialog RFC Interface)

You can use standardized dialog RFC calls and RFC calls for the following mail and workflow functions:

- **Displaying a Mail Inbox**
- **Displaying a Mail Outbox**
- **Displaying a Workflow Inbox**

See also:

Mail and Workflow Functions [Seite 536]

General Functions (Dialog RFC Interface)

You can use standardized dialog RFC calls and RFC calls for the following general functions:

- **Display product structure**
- **Display where-used list for material/document**
- **Finding Objects via Matchcode**
- **Finding Objects in Classes**
- **Requesting SAP System Parameters**

See also:

General Functions in the SAP Environment [Seite 540]

Examples of How the CAD Dialog Interface Can be Used

This section describes the data flow in some of the processes that might occur when you use an external system together with the SAP-CAD dialog interface.

These processes are divided into a series of sub-processes, each of which is shown in a diagram. The diagrams are structured as follows:

- On the **left** of the diagram, you see the individual steps that make up the process. Each box represents an activity in the external system or a function of the SAP-CAD dialog interface.
- On the **right** of the diagram, you see the function from the SAP function library for each step of the process.

The processes here are examples and cannot be taken to apply in every case. In the examples, the release procedure generally takes place in the external (CAD) system. The SAP System can be regarded as a subsequent application area.

The following processes are described:

- Finding a Document Info Record via Matchcode [Seite 51]
- Designing a Single-Part Drawing [Seite 53]
- Designing a Single-Part Drawing by Copying [Seite 55]
- Designing an Assembly Drawing [Seite 57]

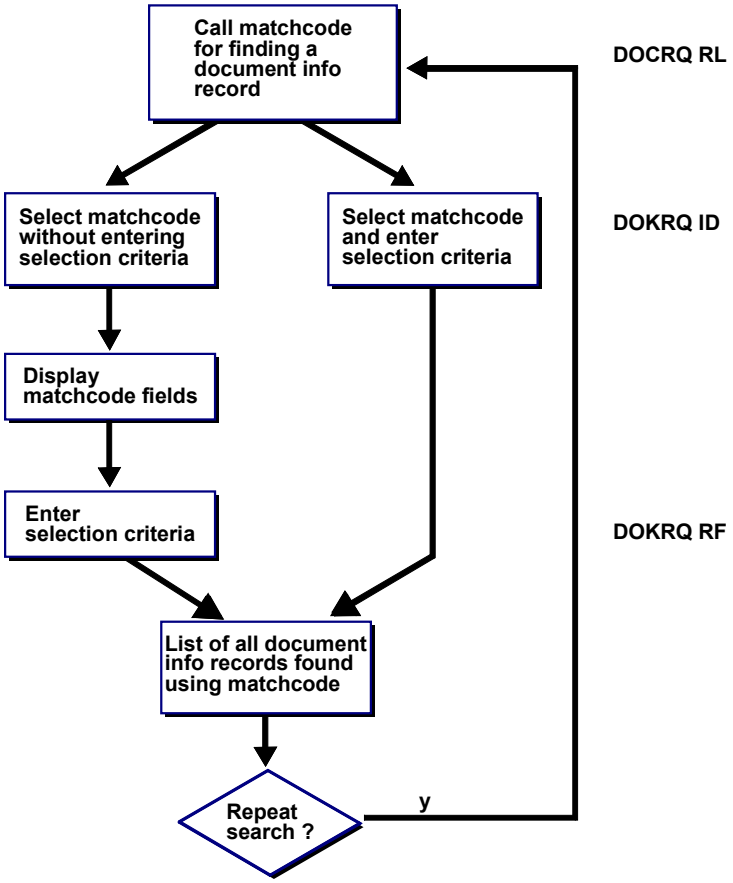
Finding a Document Info Record via Matchcode

This example shows a possible sequence of functions for finding a document info record via a matchcode.

- First, use the SAP-CAD dialog interface to display all the matchcodes for finding document info records.
- You have the following two options:
  - If you know the structure of the matchcode you required, enter the matchcode ID followed by the selection criteria for the matchcode. You then see a list of all the document info records that match your selection criteria.
  - If you do not know the structure of the matchcode you require, first select the matchcode ID. The SAP System transfers the fields of the matchcode ID to the external system. Enter your selection criteria in these fields. You then see a list of all the document info records that match your selection criteria.
- If none of the document info records in the list refers to the object you are looking for, repeat the search process using different selection criteria or a different matchcode ID.

Finding a Document Info Record via Matchcode

Finding a Document Info Record via Matchcode

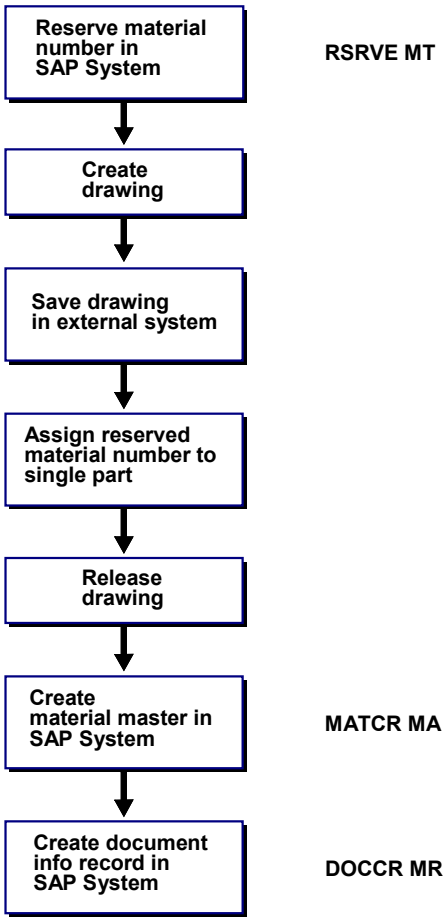


Designing a Single-Part Drawing

This example shows a possible sequence of functions for creating a drawing for a single part without copying data from an existing drawing.

- First, reserve a material number in the SAP System for the drawing object you want to create.
- Then start to work on your drawing. This process may be spread over a period of time. The drawing is stored under a drawing number in your external system.
- Once the single-part drawing is finished, assign the material number you reserved in the SAP System to the drawing. You can then release the drawing.
- Finally, create a material master record in the SAP System (with the material number you reserved), and create a document info record for the drawing you created.

Creating a Single Part Drawing

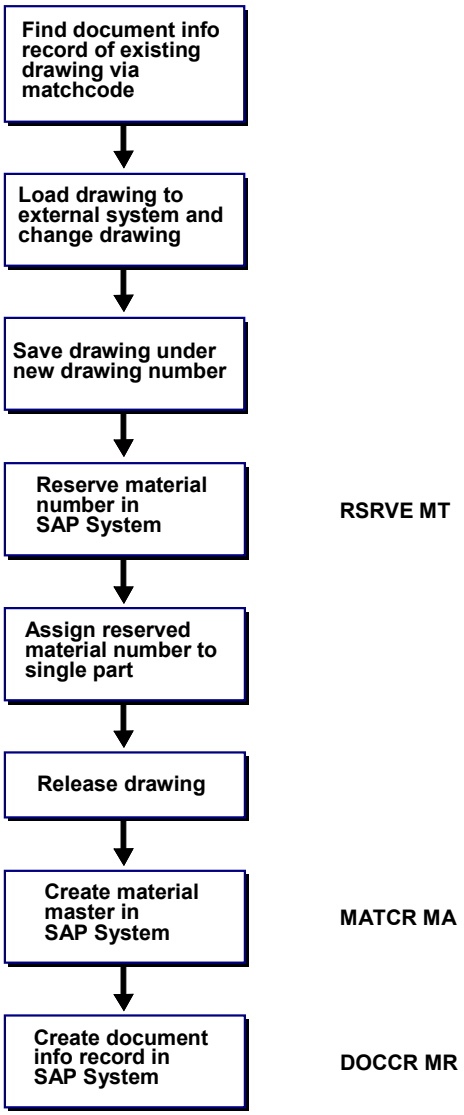


Designing a Single-Part Drawing by Copying

This example shows a possible sequence of functions for creating a drawing for a single part by copying data from an existing drawing.

- First, select the drawing you want to copy from. Provided that document info records have been created in the SAP System for all drawings, you can find a suitable drawing using a matchcode for document info records. See the section entitled “Finding a Document Info Record via Matchcode” for an example sequence of functions.
- Once you have found the document info record of the drawing you want to copy from, load the drawing to your external system.
- Change the drawing as required and save the changed drawing in your external system under a new drawing number.
- Reserve a material number in the SAP System and assign the material number you reserved to the drawing.
- Once the drawing is finished, release the drawing.
- Finally, create a material master record in the SAP System with the material number you reserved, and create a document info record for the drawing you created.

Creating a Single Part Drawing by Copying

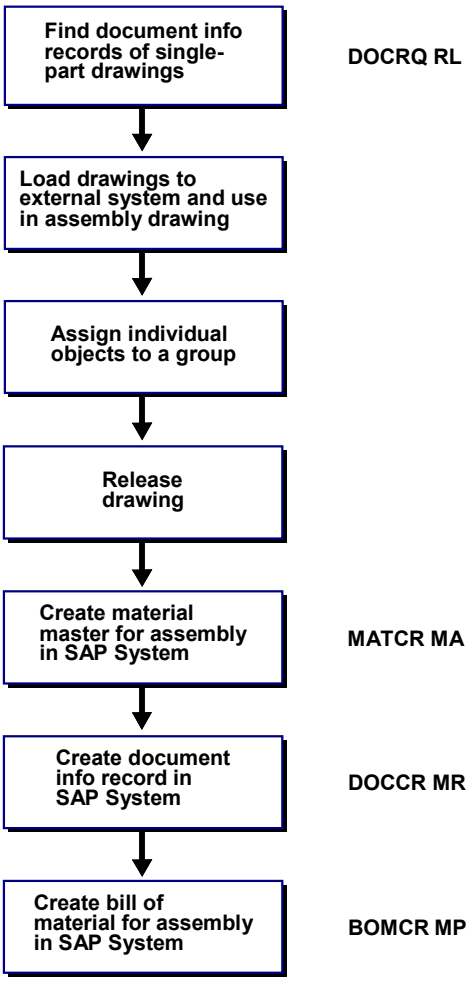


Designing an Assembly Drawing

This example shows a possible sequence of functions for creating an assembly drawing.

- First, select the single-part drawings you want to use in your assembly drawing. Provided that document info records have been created in the SAP System for all drawings, you can find the drawings you require using a matchcode for document info records.
- Once you have found the document info records of the single-part drawings you want to use in your assembly, load the drawings to your external system and put them together in one drawing. Assign the individual drawing objects to a group.
- Once you have released the drawing, use the SAP-CAD dialog interface to create a material master record for the assembly in the SAP System, create a document info record, and create a BOM for the group.

Putting Several Single Part Drawings Together to Form an Assembly Drawing



CAD Interface (CA-CAD)

CAD Interface (CA-CAD)

Introduction to the CAD Interface [Seite 15]

Technical Description of the CAD Dialog Interface [Seite 60]

Technical Description of the Dialog RFC Interface [Seite 454]

Appendix [Seite 625]

Technical Description of the CAD Dialog Interface

This section contains a technical description of the CAD interface.

General Information on the CAD Dialog Interface [Seite 61]

SAP Function Library [Seite 62]

SAP Interface Program [Seite 63]

Notes on Installation [Seite 64]

Customizing the SAP-CAD Function Library [Seite 65]

Configuration File cadrfc.ini [Seite 460]

Configuration File sideinfo [Seite 67]

Configuration File saprfc.ini [Seite 69]

Customizing the SAP System [Seite 70]

General Information [Seite 71]

Control Parameters [Seite 72]

General Data [Seite 74]

User-Defined Functions [Seite 76]

Example: Displaying a Where-Used List [Seite 77]

Example: Settings for a User-Defined Function [Seite 79]

Including RCCADCOM in User-Defined Functions [Seite 84]

Emulation of the SAP System [Seite 85]

Emulation Library [Seite 86]

Data File cademu.r3 [Seite 87]

Functions in the CAD Dialog Interface (I) [Seite 89]

Description of Function Library caddialg [Seite 310]

Functions of the CAD Toolbox Library [Seite 332]

General Information on the CAD Dialog Interface

The CAD dialog interface supports bidirectional data transfer online between your external system and your R/3 System. The SAP System functions as a server. Essential dialog is shown in special screens in the external application. In a LAN/WAN, the interface uses the TCP/IP protocol.

The interface comprises the following components:

integrated in the R/3 System: SAP Function Library [Seite 62]

on the technical hardware platform: SAP Interface Program [Seite 63]

The CAD interface enables you to transfer data online for the following application areas between an external system and an SAP System:

- Document management
- Material master records
- Bills of material
- Engineering change management
- SAP mail and workflow
- Plant maintenance
- Classification
- Configuration

SAP Function Library

The SAP function library is a collection of sub-programs written in the programming language **C**.

Each of these sub-programs implements a specific function when communicating with the SAP interface program. Before you can access the individual functions from your external system, the function library must be installed in the engineering application. To do this, an application-specific program must be installed in the application.

Most functions can be configured according to user requirements when you install your system, using SAP tables. For example, you can define which data is transferred in which order. You can also define the length of fields for transfer, which may be different from the field length defined in the R/3 Repository.

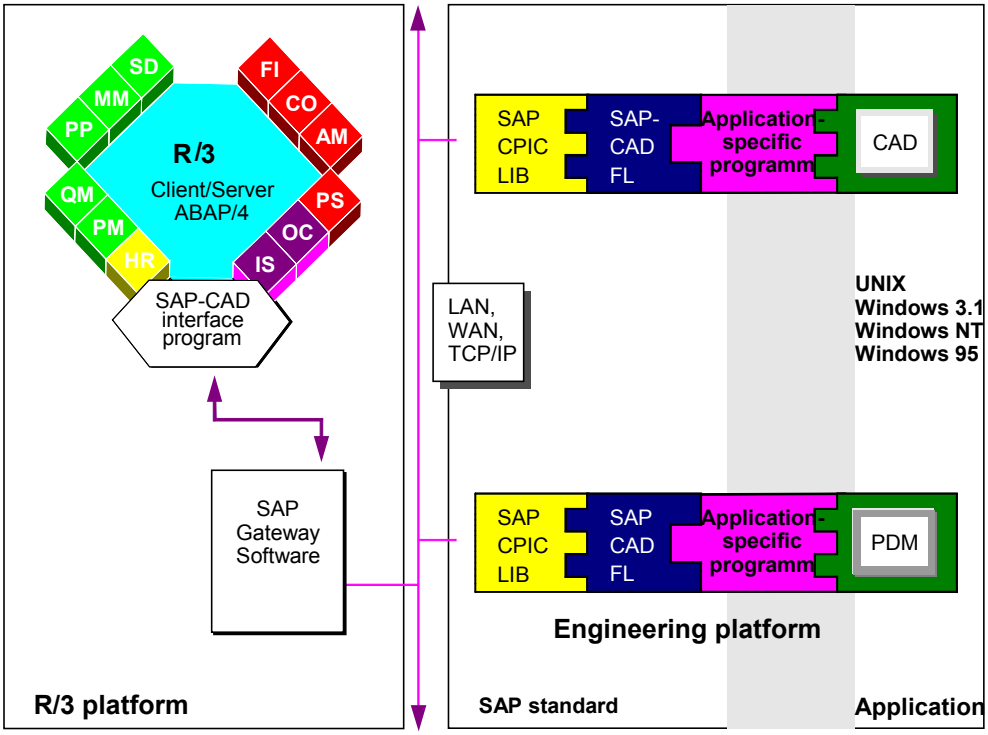
SAP Interface Program

The SAP interface program fulfils the following functions:

- The program receives commands from the external system, reads the data requested in the SAP database, and converts this data to the format required for transfer. This data is then transferred to the external system. If the data requested can be changed in the external system, the data is locked in the SAP System.
- The program checks the data received from the external system for consistency, then transfers the data to the appropriate SAP transaction so that the database can be updated.
- If the SAP database is changed as a result of data transferred from the external system, the program ensures that the same data is changed as would normally be changed in the online SAP transaction.

The figure below shows the data flow via the CAD interface.

Data flow via the CAD interface



Notes on Installation

The workstation-specific part of the software is available as standard for the following operating systems:

- DEC OSF/1
- HP UX
- IBM AIX
- SNI SINIX
- Sun Solaris
- MS Windows 3.1
- MS Windows NT
- MS Windows95 (software identical to MS Windows NT)

You will find the software on the SAP CD *Presentation with SAP-GUI*.

For Unix platforms, the software is stored as a compressed file called *cad.car* in the relevant platform sub-directories. You can decompress the file using the SAP utility program *car*.

For MS Windows 3.1, MS Windows NT and MS Windows95, you can install the software using an additional option of the SAP GUI installation program.

After installation, see the file *cadread.me* for information on the files supplied, as well as the latest information on the current version of the interface.

To install the software in your own applications, you also require the CPIC or RFC library to communicate with R/3. These are a component part of the RFC SDK, and can also be found on the SAP CD *Presentation with SAP-GUI*.

Customizing the SAP-CAD Function Library

For information on customizing the SAP-CAD function library, see the following topics:

Before You Use the SAP-CAD Function Library [Seite 66]

Configuration File cadrfc.ini [Seite 460]

Configuration File sideinfo [Seite 67]

Configuration File saprfc.ini [Seite 69]

Before You Use the SAP-CAD Function Library

Before you can use the SAP-CAD function library to transfer data between the external application and the R/3 System, the following requirements must be met:

- A SAP Gateway process **must** be started. This happens automatically when you start the standard R/3 System.
- In local file *services,* the entries for *sapdpxx* and *sapgwxx* must be maintained. The characters xx represent the R/3 System number.

sapgw00	3300/tcp
sapdp00	3200/tcp

- Configuration file *cadrfc.ini* must be maintained

Configuration File *sideinfo*

File *sideinfo* is used to define the partner program (SAP System) you want to communicate with. You can make entries for several systems in this file.

As of Release 4.0A, this file will only be used if the connection is to be established with an R/2 System. If you want to establish a connection to an R/3 System, use file *saprfc.ini.*

See also:

File sideinfo [Seite 635]

Each entry has the following structure:

- **DEST**

  Logical name of the SAP System

  This is the name defined in configuration file *cadrfc.ini* or that has been entered as a parameter with the function *SapConnc.*

- **LU**

  Name of the SAP System logical unit (LU) that is defined in the SNA software.

- **TP**

  Entry is not required with R/2.

- **GWHOST**

  Name of Gateway computer

  This is usually the computer on which the R/3 System is started.

- **GWSERV**

  Entry in form *sapgwxx* (xx is the Gateway process number)

  A matching entry must exist in local file *services.*

- **PROTOCOL**

  Transfer protocol

  Must always be **C** for R/2.

- **PIC_TRACE**

  You can use this parameter to switch the extended CPIC trace (trace file name *CPxxxx.TRC*) on or off.

  The following trace levels are supported:

0	Trace off
1-3	Trace on (1 is the lowest level of detail, 3 is the highest level of detail)

Configuration File saprfc.ini

As of Release 4.0A, configuration file *saprfc.ini* is used instead of *sideinfo* when you want to establish a connection to an R/3 System.

You can make entries for several systems in this file. For more information, refer to File saprfc.ini [Seite 636]

Customizing the SAP System

The following topics contain hints on customizing the SAP System:

General Information [Seite 71]
Control Parameters [Seite 72]
General Data [Seite 74]
User-Defined Functions [Seite 76]
Example: Displaying a Where-Used List [Seite 77]
Example: Settings for a User-Defined Function [Seite 79]
Including RCCADCOM in User-Defined Functions [Seite 84]

General Information on Customizing

Before you can use the CAD interface, you need to define settings in the customizing functions. You define control parameters and the structure of data strings for transfer in individual functions.

SAP supplies the CAD interface with default settings for the individual functions. You can base your settings on these standard settings, and change them to suit your requirements. The default settings are supplied in client 000. You can copy these settings to the client you are working in by choosing *Utilities* → *Compare* → *Client 000* in the function for customizing the CAD interface.

The following topics describe the actions you need to take when customizing the SAP System. There are also examples of settings for individual customizing functions.

The Customizing menu includes the following functions:

- Control Parameters [Seite 72]
- General Data [Seite 74]
- User-Defined Functions [Seite 76]

Control Parameters

The control parameters are general parameters that define how the CAD interface works. We make a distinction between control parameters and debug parameters. You can define the following parameters:

- **Maximum number of matchcode records**:

  This parameter defines the maximum number of matchcodes after which communication with the external application program is resumed when you find objects via matchcode. You can make further function calls to request more matchcode records.

- Separator in data strings:

  This parameter defines the separator character used in transferred data strings. This character is used in all situations where the structure of a data string cannot be defined in customizing functions. The field itself is for display only. To maintain the character in this field, enter a hexadecimal code in the next field *(Hex code).*

- **(Hex code)**:

  This is the entry field for the hexadecimal code that defines the separator character used in transferred data strings. You can use the F4 possible entries function to display a list of codes. The character that the code represents is displayed in the *Separator in data strings* field.

- **Sort BOM item**:

  This parameter defines how transferred BOM items without item numbers are sorted when you first create a BOM or when you change a BOM.

- **Maximum length of send string**:

  This parameter defines the length of data strings that can be transferred between the external system and the SAP System. If the data string you want to transfer is longer than this parameter allows, several data strings are sent. These strings are concatenated to form a single data string again after transfer.

- **External assignment**:

  This indicator is supported for all objects that can be created in the SAP System via the CAD interface. The indicator determines whether external object number assignment is allowed or not.

- **Internal assignment**:

  This indicator is supported for all objects that can be created in the SAP System via the CAD interface. The indicator determines whether internal object number assignment is allowed or not.

- **Reserve material number**:

  This parameter switches the option for reserving a material number on and off.

- **Material type**:

  If the option for reserving a material number is active, you can define one material type as a default value for determining the number range.

The control parameters below are only relevant to **test** or **debug mode** for the CAD interface:

- **Transaction modes**:

  These parameters define the processing mode for the individual SAP transactions that can be called via the CAD interface. They only apply to the transactions if you start the CAD interface in test mode via report RCCADI10. If you start the CAD interface from an external system, these parameters are not relevant.

- **Line length of standard text**:

  If you start the CAD interface in test mode via report RCCADI10, the data that is usually sent from the external system is read from a standard text. This parameter defines the number of characters in the standard text that are read. When you create a standard text, you need to keep to this length.

- **Line length for screen display**:

  In test mode for the CAD interface, the data strings that are usually transferred between the external system and the SAP System are listed on screen. This parameter defines the number of characters that are displayed on the screen. The data strings are wrapped dynamically.

- **TRACE on**:

  If the TRACE is switched on, the data strings transferred between the external system and the SAP System are logged in table INDX. You can analyze the log afterwards using report RCCADI20. This allows you to automatically transfer the logged function calls to a standard text. This filters out the data strings sent from the SAP System to the external application program, so that these strings are ignored. Finally, you can start the CAD interface in test mode using report RCCADI10, in order to analyze the problems that occurred in detail.

Customizing General Data

The following steps allow you to configure the individual functions that can be called via the CAD interface:

- *Define functions*
- *Allocate field sets*
- *Define local field names*
- *Define field sets (SAP->CAD)*
- *Define field sets (CAD->SAP)*

Step *Define functions*

In this step, you enter all the function calls you want to use via the CAD interface. If you do not enter a CAD interface function here, you cannot call the function. Select the F4 possible entries function on the *Process code* field to see a list of all the functions supported by the CAD interface. If you select the F4 possible entries function on the *Sub-process code* field, you see a list of all the function calls for a process code.

Do **not** enter user-defined functions that are created as enhancements to the standard CAD functionality in this step.

Step *Allocate field sets*

In this step, you allocate a field set to each function call for which you can define the structure for transferred data strings. Do not enter functions that have a predefined structure for the transferred data string here.

All CAD interface functions that have the same field set transfer the same data fields. The allocation of field sets is also dependent on the external system, so you can allocate field sets to a CAD interface function such that different data is transferred for different external systems. You can use the F4 possible entries function to see a list of all the CAD interface functions to which you need to allocate a field set.

Step *Define local field names*

In this step, you define the link between the local field names in the external application program and the SAP screen fields. This step is required in order to transfer data from the external system to the correct fields in the SAP System.

Before you create this link, we advise you to familiarize yourself with the dialog of the SAP transaction concerned. The F4 help function can be useful here. If you use `F4` on the *Screen number* field, you see a list of the functions entered in step *Allocate field sets.* You only define local field names for these functions. If you select a function, you see a list of all the screens for this function. Select a screen to see a list of entry fields on this screen. Select the field you require. The entire entry is copied to the customizing table. If you enter a screen name before you call `F4`, you see a list of all the entry fields on this screen.

When you make an entry, the database length, the conversion routine (if any), and the check table (if any) are displayed.

Step *Define field sets*

In the steps *Define field sets (SAP->CAD)* and *Define field sets (CAD->SAP),* you define which fields belong to the field sets entered in step *Allocate field sets.*

In step *Define field sets (SAP->CAD),* you process the field sets that have send direction *'D'.* This means that the data string is sent from the SAP System to the external system.

In step *Define field sets (CAD->SAP),* you process the field sets that have send direction *'U'.* This means that the data string is sent from the external system to the SAP System.

In both steps, enter the name of the field set. You can use the F4 help function to see a list of field sets defined for the send direction concerned (in step *Allocate field sets)*. Then enter a consecutive number to distinguish different entries for a field set from each other. You also need the local field names used in the external application program.

- In step *Define field sets (CAD->SAP),* the field name must be defined as a local field name first. You can use the `F4` help function to see a list of the local field names that are relevant to the context.

  You can also enter a default value and a field length. The default value is entered automatically whenever no value is transferred from the external application program. The entry in the *Field length* field overrides the field length in the R/3 Repository.

- In step *Define field sets (SAP->CAD),* you can use any local field names. Since these field names do not need to be predefined, you cannot use the `F4` help function here.

  Enter the values for the SAP table field name and the field length. You can use the `F4` help function to see a list of the table field names you can use. You have the option of making an entry in the *Field length* field. This entry overrides the field length in the R/3 Repository.

User-Defined Functions

Customizing for the standard functions of the CAD interface is described in the previous topics. If you want to enhance the functionality of the CAD interface by creating user-specific functions, you need to define certain settings in the step *Define user-defined functions.*

A user-defined function can comprise several communication steps. This means that several entries may belong to the same function.

- Enter the same number for all entries belonging to a function.
- They are distinguished by the value in the *Step* field, where a consecutive number is entered.
- Next, enter a value for he process code.

  Process codes *USR01, USR02,... USR10* are available for user-defined functons.
- There is no restriction on sub-process codes.
- Enter either *'U'* or *'D'* for the send direction, depending on whether the communication step transfers data from the external system to the SAP System ('U') or the other way round ('D').
- For communication steps with send direction *'U',* enter a value for the *Program* and *FORM routine*.

  The control program for the CAD interface then automatically goes to the correct FORM routine in the program you enter. The transferred data is made available to the FORM routine and can be analyzed in table CADIN.

Please note that you cannot use next record calls in user-defined functions. This is because routines must be defined in the communication modules for these functions. However, due to the general character of user-defined functions, these routines cannot be made widely available.

Example: Settings for Displaying a Where-Used List

This topic takes you through a customizing function, using function *BOMWU RL (Display where-used list)* as an example. For information on how to create a user-defined function, please refer to the topic following after this one.

- To enable you to display a where-used list via the CAD interface, enter function *BOMWU RL* in step *Define functions.* Then enter a field set containing the data structure for transfer in both send directions under step *Allocate field sets.* In both case, the field set used is **WH_USED.** The name of the external system used is **DEMO**.
- Next, allocate data fields to both field sets. First, we will deal with the field set used for send direction 'U'. Before you can define the structure for this field set, the local field names used in the external system must be declared in step *Define local field names.* To see which data fields need to be transferred to the SAP System for a where-used list, run through the SAP transaction online.

  Use the menu path *Logistics* → *Production*→ *Master data* → *Bills of material* to get to the BOM menu. You find the where-used list function for materials under *Reporting* → *Where-used list* → *Material.* You can use function *BOMWU RL* to display a where-used list for materials, documents, or classes via the CAD interface. However, this example concentrates on the where-used list for materials.

  When you run the online transaction, you see that you need to transfer the number of the material for which you want to see a where-used list. You can also transfer indicators that define the type of where-used list (direct or via classes), and the category of BOM in which you want to find the material (equipment BOM, material BOM, and so on). You can also enter two dates to define the effectivity period. On the next screen, you can make further entries that define the required quantity, the plant, the item category, the BOM usage, and whether you want a multi-level where-used list. Do not set the *Long text* indicator, because no long texts can be transferred via the CAD interface for this function. The settings for display and print formats are not relevant, because these only affect the format in online mode.

  Once you know which fields you need, you can define the local field names in step *Define local field names.*

  In this example, the following fields are to be transferred:

Technical field name	Field description
MATERIAL	Material
DIRECT	Indicator: *direct where-used list*
VIA_CLASS	Indicator: *where-used list via classes*
MAT_BOM	Indicator: *use in material BOMs*
EQUI_BOM	Indicator: *use in equipment BOMs*
VALID_FROM	*Valid-from date*
VALID_TO	*Valid-to date*
BOM_USAGE	*Use*

Example: Settings for Displaying a Where-Used List

These are the local fields we want to use. Enter the names in step *Define local field names.*

If you want to enter, for example, the local field MATERIAL, you can allocate the field to the corresponding SAP screen field as follows: select F4 help to see a list of all the functions entered in step *Allocate field sets* that have send direction 'U'. These are the fields for which you define local field names. In the list, double-click on the plus sign before entry *BOMWU RL* to expand it.

You then see a list of the screens allowed for this function. The field for the material is on screen SAPMM29L0200 (*Material Where-Used List: Initial Screen*). Explode this entry to see a list of all the entry fields defined for this screen. The checkboxes before the fields show whether the field has already been selected. The field we want is field RC29L-MATNR. Double-click on this line to copy the entire entry to the customizing table.

Proceed in the same way for the other fields.

- Now you can define the field set WH_USED for function call BOMWU RL in step *Define field sets (CAD->SAP).* You can use the F4 help function on the *Field set* field to see a list of all the functions entered in step *Allocate field sets* that have send direction 'U'.

  Select field set WH_USED, or enter the field set directly. Enter a consecutive position number that defines the sequence of the fields in the field set. In the *Field name* field, enter a local field name. You can use the F4 help function to see a list of the local field names that can be used in the current field set. The checkboxes before the fields show whether the local field name is already used in the field set. Double-click on a field that is not yet used to copy the entire entry to the customizing table.

- Once you have defined the field set for send direction 'U', define the second WH_USED field set in step *Define field sets (SAP->CAD).* This field set is for transferring the resulting where-used list to the external application program.

  First choose step *Define field sets (SAP->CAD).* You can use the F4 help function to see a list of all the functions entered in step *Allocate field sets* that have send direction 'D'. Either select or enter field set WH_USED. Enter a consecutive position number that defines the sequence of the fields in the field set. In the *Field name* field, enter a local field name. However, since you do not need to declare local field names for send direction 'D' in step *Define local field names,* you cannot use the F4 help function on this field. You can enter any value.

  In the *DB field name* field, enter the SAP table field name that corresponds to the local field name. You can use the F4 help function as described above to see a list of the SAP tables allowed for the field set entered. You can select fields from these tables. Double-click on the plus sign by the table to display all the table fields in the table. The checkboxes before the fields show whether the field is already used in the field set.

Once you have entered all the fields you require in this way, the customizing process for function *BOMWU RL* is complete. You can now call this function via the CAD interface.

Example: Settings for a User-Defined Function

This topic describes how to create a user-defined function. The function in this example is for shifting the valid-from date of a change number. The standard CAD interface functionality does not include a function that does this but you can change the valid-from date using the standard function *CHGCH MR* (change change master record). However, this only works for change numbers that have not yet been used to make changes. As soon as a change number has been used to make a change, you need to use a special function to change the valid-from date.

Data of the SAP transaction

Before you can program a function as a CAD interface function, we advise you to look at the SAP transaction you want to call - in this case CC02 *(Change change master)* - in the online SAP System. To call this function, choose the menu path *Logistics* → *Central functions* → *Engin. change mgmt* → *Change master* → *Change*. To implement the function, you need to know certain technical information, such as the number of a screen or the field name of an entry field. To find this information, call the F1 help function on the screen field concerned, and select *Technical info*.

Enter an existing change number and confirm your entry. You see the main screen of the change master record. If changes have already been made using the change number, you see that the *Valid from* field is no longer available for entry. You can only change the validity date by choosing *Edit* → *More* → *Shift date.*

To set up batch input, you need the function code to be entered on the main screen. To see a list of function key definitions, click with the right-hand mouse button. You see the number of the function key that triggers the function you require. However, this only works with functions for which function keys are defined.

Functions that you reach via the menu cannot be displayed like this. To see their internal function code, you have to go to the debug mode using the function code '/h'. Then select the function you require via the menu. You see the program in debug mode. Display the content of system field *SY-UCOMM*, which contains the value for the internal function code. Select *Continue* to exit debug mode and resume processing.

In this case, the procedure described above gives you function code DATU. You enter this code on the main screen to access the date shift function. You see a dialog box with screen number SAPMC29C 0130, on which the date field is available for entry. You can enter the new date, then select *Copy* to return the the main screen, where you can save your change.

Design for the CAD Interface Function

Following through the transaction, you reach the following design for the CAD interface function for the date shift. The change number and the new valid-from date is sent from the external system. This data is used to construct a batch input for transaction CC02. This batch input should then call the DATU function in the transaction.

No other fields should be able to be changed apart from the date field. Because defining field sets for user-defined functions in the CAD interface is not supported, a separator is used to separate the change number and the new date in the send string. Function call *USR01 SD* identifies this function. The data string sent from the external system with this call contains the change number and then, separated by a separating character, the new date. The date format that is defined in your user master record should be used. This avoids conversion problems.

Example: Settings for a User-Defined Function

Program

We will call the program for encoding the function RCCADCHG. First create this program. Then encode the function as follows:

```
REPORT RCCADCHG.
TABLES: RC29A.
INCLUDE RCCADCOM.
*---------------------------------------------------------------------*
*       FORM SHIFT_DATE_OF_CHANGE_NUMBER                              *
*---------------------------------------------------------------------*
*       Die Routine verschiebt das Datum der angegebenen              *
*       Änderungsnummer auf den neuen Wert.                           *
*       Der empfangene Datenstring hat ein vorgegebenes Format        *
*       und ist wie folgt aufgebaut:                                  *
*                                                                     *
*       -         Änderungsnummer                                     *
*       -         neues Änderungsdatum                                *
*                                                                     *
*       Die Elemente des Strings sind jeweils durch genau ein         *
*       Byte getrennt, das den Wert des Feldes TCIS-DELIM ent-        *
*       haelt. Dieser Wert wird auch als Ende-Satz-Kennzeichen        *
*       verwendet.                                                    *
*---------------------------------------------------------------------*
*       Keine USING-Parameter                                         *
*---------------------------------------------------------------------*
FORM SHIFT_DATE_OF_CHANGE_NUMBER.
DATA: CC02_MODE(1)  TYPE C,
        CHANGE_NUMBER LIKE RC29A-AENNR,
        NEW_DATE(10)  TYPE C.
FIELD-SYMBOLS: <DATA_STRING>.
CLEAR: ND_OCC_LENGTH, ND_STRING, RC29A.
CLEAR: MSGTY, MSGID, MSGNO, MSGV1, MSGV2, MSGV3, MSGV4.
CLEAR: CADOUT.      REFRESH CADOUT.
  CLEAR: DYNPRO_FTAB. REFRESH DYNPRO_FTAB.
* Fehler, wenn keine Daten gesendet wurden
  IF HEADER-DATLN EQ 0.
```

```
    MOVE: 'E' TO MSGTY, 'CH' TO MSGID, '006' TO MSGNO.
CALL FUNCTION 'WRITE_MESSAGE'
                       EXPORTING MSGTY = MSGTY
                                 MSGID = MSGID
                                 MSGNO = MSGNO
                                 MSGV1 = MSGV1
                                 MSGV2 = MSGV2
                                 MSGV3 = MSGV3
                                 MSGV4 = MSGV4
                                 MSGV5 = SPACE
IMPORTING ERROR = ERROR_FLAG
                                 MESSG = MESSAGE
                                 MSGLN = MESSAGE_LENGTH.
  ENDIF.
CHECK ERROR_FLAG IS INITIAL.
* Die Tabelle CADIN enthält die gesendeten Daten
  ASSIGN CADIN+HEADER_LENGTH(HEADER-DATLN) TO <DATA_STRING>.
* Separieren von Änderungsnummer und Datum aus dem Datenstring
  SPLIT <DATA_STRING> AT TCIS-DELIM INTO CHANGE_NUMBER NEW_DATE.
* Konvertieren der Änderungsnummer in Großbuchstaben
  TRANSLATE CHANGE_NUMBER TO UPPER CASE.
* Aufbauen der Tabelle DYNPRO_FTAB für den Batch-Input
  CLEAR: DYNPRO_FTAB.
* Einstiegsbild
  MOVE: 'SAPMC29C'     TO DYNPRO_FTAB-PROGRAM,
        '0100'         TO DYNPRO_FTAB-DYNPRO,
        'X'            TO DYNPRO_FTAB-DYNBEGIN,
        'RC29A-AENNR'  TO DYNPRO_FTAB-FNAM,
         CHANGE_NUMBER TO DYNPRO_FTAB-FVAL.
APPEND DYNPRO_FTAB.
* Hauptbild -> anwählen der Funktion 'Datum verschieben'
  CLEAR: DYNPRO_FTAB.
MOVE: 'SAPMC29C'   TO DYNPRO_FTAB-PROGRAM,
        '0110'       TO DYNPRO_FTAB-DYNPRO,
        'X'          TO DYNPRO_FTAB-DYNBEGIN,
```

Example: Settings for a User-Defined Function

```
        'BDC_OKCODE' TO DYNPRO_FTAB-FNAM,
        'DATU'       TO DYNPRO_FTAB-FVAL.
APPEND DYNPRO_FTAB.
* POP UP zur Eingabe des neuen Datums
  CLEAR: DYNPRO_FTAB.
MOVE: 'SAPMC29C'    TO DYNPRO_FTAB-PROGRAM,
        '0130'        TO DYNPRO_FTAB-DYNPRO,
        'X'           TO DYNPRO_FTAB-DYNBEGIN,
        'RC29A-DATUV' TO DYNPRO_FTAB-FNAM,
         NEW_DATE     TO DYNPRO_FTAB-FVAL.
APPEND DYNPRO_FTAB.
CLEAR: DYNPRO_FTAB.
MOVE: 'BDC_OKCODE' TO DYNPRO_FTAB-FNAM,
        '/5'         TO DYNPRO_FTAB-FVAL.
APPEND DYNPRO_FTAB.
* Hauptbild -> sichern
  CLEAR: DYNPRO_FTAB.
MOVE: 'SAPMC29C'   TO DYNPRO_FTAB-PROGRAM,
        '0110'       TO DYNPRO_FTAB-DYNPRO,
        'X'          TO DYNPRO_FTAB-DYNBEGIN,
        'BDC_OKCODE' TO DYNPRO_FTAB-FNAM,
        '/11'        TO DYNPRO_FTAB-FVAL.
APPEND DYNPRO_FTAB.
* Setzen des Abspielmodus der Transaktion
  IF DATA_SOURCE EQ 'CAD'.
    MOVE: 'N' TO CC02_MODE.
  ELSEIF DATA_SOURCE EQ 'STX'.
    MOVE: 'A' TO CC02_MODE.
  ENDIF.
* Aufruf der Transaktion CC02
  CALL TRANSACTION 'CC02' USING DYNPRO_FTAB MODE CC02_MODE UPDATE 'A'.
* Übernehmen der Nachricht aus der Transaktion
  CALL FUNCTION 'WRITE_MESSAGE'
                     EXPORTING MSGTY = SY-MSGTY
                               MSGID = SY-MSGID
```

```
                                MSGNO = SY-MSGNO
                                MSGV1 = SY-MSGV1
                                MSGV2 = SY-MSGV2
                                MSGV3 = SY-MSGV3
                                MSGV4 = SY-MSGV4
                                MSGV5 = SPACE
IMPORTING ERROR = ERROR_FLAG
                                MESSG = MESSAGE
                                MSGLN = MESSAGE_LENGTH.
* Füllen der Tabelle CADOUT mit der Nachricht und senden
  PERFORM ASSEMBLE_NORMAL_OUTPUT(SAPMCCAD) USING INIT_PCODE INIT_UCODE
                                                 ND_STRING
                                                 ND_OCC_LENGTH SPACE.
ENDFORM.
```

Including RCCADCOM in User-Defined Functions

Program RCCADCOM must be included in all user-defined functions. This program contains the definition of the CAD interface data fields that must be present in user-defined programs. For example, this program contains internal tables CADIN and CADOUT, which are used to process data transfer between the external system and the SAP System.

The control program for the CAD interface uses table CADIN to make the data sent from the external system available to the user-defined program. Batch input for transaction CC02 is set up with the data received. Once the transaction has been called, the message output by the transaction is written to table CADOUT and sent to the external application program. A routine from the CAD interface is used to write data to table CADOUT via an external perform call.

Before you can access the user-defined function via the CAD interface, you need to make the following two customizing entries under *Define user-defined functions:*

```
001   01   USR01   SD   U   RCCADCHG   SHIFT_DATE_OF_CHANGE_NUMBER

001   02   USR01   SD   D
```

This defines the communication flow for the user-defined function. For each function call with send direction ‘U’, enter a program name and a FORM routine. This routine is then accessed dynamically when you call the function.

This procedure allows you to enhance the standard functionality of the CAD interface by adding any functions you require.

Emulation of the SAP System

In order to implement an SAP connection to an external system using the CAD dialog interface, you must have access to an SAP System.

Since this is not always feasible, SAP gives you the option of developing an interface independently of any SAP System by using a file interface.

The following topics describe this option:

Emulation Library [Seite 86]

Data File cademu.r3 [Seite 87]

Emulation Library

The SAP emulation functionality is based on the C library *cadcpemu.o* or CADCPEMU.LIB. This library must be installed in the external application in place of the CPIC library.

As a result, the SAP function calls are not transferred to an SAP System. Instead, the function calls attempt to read a response record from an ASCII file for each request.

This emulation supports all function calls of the CAD dialog interface, with the exception of the function calls for document management that are used to check in and check out original application files.

Data File *cademu.r3*

Before you can use the SAP emulation functionality, the directory from which you start the application-specific interface program must contain a file called *cademu.r3*, which must have a specific structure. A file containing data for certain function calls is supplied as standard with the interface software. This file can be used as a basis for your own developments and can be changed as required.

The structure of the data strings stored in file Datei *cademu.r3* is the same as the structure defined in the customizing settings supplied as standard by SAP. These settings are shown in the emulation file as the response strings for function call SYSRQTB.

Please note the following rules for modifying the settings:

- Data strings that are to be interpreted as sent by the function call must be entered in the file in left-justified format.

MATRQMR	53	EB1024	0101

- All data strings entered in this format are checked to see if they match the data string sent. The comparison covers up to 60 bytes. If you have trouble getting the entry in file *cademu.r3* to match the data string sent, please check log file *log_xxxx.log* to see if your send string entry has the right structure.
- If there are several identical entries, the first that matches is selected.
- All response records must start with the following three special characters: **blank

  The line length for response records must be 60 characters, including the three special characters. A response can run over as many lines as you like.
- A dividing line must be inserted before the first line and after the last line of each record.
- If several response records are sent for a function call, all the response records follow on from each other, separated by a dividing line. For an example, see function call MATRQMC in file *cademu.r3*.
- If you want to enter function calls for which no example entry exists, you need to know the header structure of the response string.

  The header has a fixed length of 19 bytes. The header must be at the start of each dataset that is to be sent or that has been sent, and has the following structure:

Header element	Byte
Process code	5
Sub-process code	2
Returncode	2
Message length	3
Data length	6
Next record indicator	1

Data File cademu.r3

The message sent by the SAP System, if any, is placed immediately after the header. The actual data follows after this.

You are working in emulation mode, so you must set the *CadDialogEmulationMode* parameter in the *cadrfc.ini* configuration file to 1.

General Functions in the CAD Dialog Interface

This section contains a description of the first part of the functions in the CAD dialog interface. For some of the functions, there are examples that clarify the communication procedures.

General Information [Seite 90]

Functions

General Functions (CAD Dialog Interface) [Seite 93]

Functions for Material Master Records [Seite 110]

Functions for Document Management [Seite 138]

BOM Functions [Seite 174]

Functions for Engineering Change Management [Seite 186]

Functions for Plant Maintenance [Seite 236]

Classification System Functions [Seite 274]

Office Functions [Seite 296]

Library

Description of Function Library caddialg [Seite 310]

Functions of the CAD Toolbox Library [Seite 332]

General Information

A CAD interface function is generally made up of several communication steps that are defined using sub-functions.

- A CAD interface function is identified by a five-character process code that remains unchanged throughout all communication steps.
- An individual sub-function is identified by a two-character sub-process code.
- When you call a CAD interface function in the function library from an external system, communication takes place with the SAP interface program. The transferred data is processed in the SAP System.
- If an error situation occurs during processing in the SAP System, this usually terminates the entire function. However, there are some special cases, notably in classification functions, where the current function is not terminated. In these cases, an error dialog with the external application program is carried out.

See also:

Data Transfer Using Data Strings [Seite 91]

Data Transfer Using Data Strings

Data is transferred between the function library and the SAP-CAD interface program by transferring data strings. Unless otherwise stated, the structure of the data string transferred in a communication step is not predefined. You can use the customizing functions for the CAD interface to define the structure of data strings.

However, in some CAD interface functions, notably in classification functions, the structure of the data string transferred is predefined. This is mentioned in the description of the individual functions concerned.

Structure of data strings

The data string transferred between the external application program and the SAP System during a CAD interface function has a structure that is common to all interface functions. The string has a header section followed by a message section and a data section.

The length of the **header section** is 19 bytes. The table below shows the structure of the header section:

Structure of the header section

Structure element	Bytes	Description
Process code	5	This contains the identifier of the CAD interface function.
Sub-process code	2	This contains the identifier of the sub-function.
Returncode	2	This contains the return code set in the SAP interface program.
Length of message section	3	This indicates the length of the message section.
Length of data section	6	This indicates the length of the data section.
Next record indicator	1	This indicator shows whether another function call follows the current function call or not. If another function call follows, the send direction is changed. If not, the send direction remains the same.

The structure of the data section in a CAD interface function can follow either of the conventions described below:

- If the structure of the data string transferred is defined in customizing tables, the individual data fields are concatenated in the sequence defined and with the length defined with no gaps in between.

  Before the data strings transferred can be processed correctly, you need to synchronize the two systems. There is a special function in the CAD interface that allows you to synchronize the table settings in the external system and the SAP System at any time.

- If the structure of the data string transferred is predefined, you use separator characters to separate the individual data fields from each other. You can define the separator character in the customizing functions for the CAD interface.

General Functions in the CAD Interface

The CAD dialog interface supports the following general functions:

CONNC - Establishing a Connection to the SAP System [Seite 94]

DISCO - Terminating a Connection to the SAP System [Seite 95]

BREAK - Terminating the Current Function [Seite 96]

SYSRQ - System Functions [Seite 97]

HLPRQ - Help Functions [Seite 103]

CONNC - Establishing a Connection to the SAP System

This function is used to establish a connection from an external application program to the SAP System. The connection is maintained until you terminate it using the *DISCO* function. Because of the simple structure of this function, no sub-process code is required. The communication flow for function *CONNC* and the data transferred are shown in the table below.

If the connection to the SAP System is established successfully, the SAP interface program sends a message containing data that identifies the SAP System.

Communication flow for the CONNC function

SP code	Direction	Data transferred	Length
	→	External system identifier	10
		Interface version	5
		DISPLAY variable	20
	←	Message	variable

You can enter a local DISPLAY or HOSTNAME address as of release 2.2. This address is used in some document management functions, for example, to determine the front end-dependent access path of a document. You can also enter this parameter in the CAPRFL.INI (CardDialogNetAddress) data.

When you logon to the SAP System, the SAP-CAD interface program checks whether it is possible to work properly with the CAD interface. An attempt to establish a connection is rejected if the entry is missing from table TCIS or the entry for function *CONNC* is missing from table TCIC.

Communication example:

```
CONNC        000035 DEMO      3.0A hw1218:0.0
```

```
CONNC  00071000000 SCH001 Connection to SAP System 'S11 ' Rel. '30A
'created
```

This function is called from the external application program as follows:

```
hd = SapConnc(argv, sapsys, client, cadsys, user, passw, langu, vers,
msg)
```

DISCO - Terminating a Connection to the SAP System

This function is used to terminate a connection to the SAP System that you established previously using the *CONNC* function. You can call this function at any time. You can even call this function between two communication steps of any other CAD interface function. If you do this, the SAP-CAD interface program ensures that the current function is terminated correctly before closing the connection to the external application program.

As for function *CONNC,* no sub-process code is required for function *DISCO*. Since this function call disconnects the SAP System, the SAP System does not send a message to the external application program afterwards.

Communication flow for function DISCO

SP code	Direction	Data transferred	Length
	→	None	

This function is called from the external application program as follows:

```
hd = SapDisco().
```

BREAK - Terminating the Current Function

The *BREAK* function is used to terminate a function that you called previously and that consists of several communication steps.

- You can terminate any CAD interface function with the *BREAK* function at any point in the process flow when the external application is to send data.
- You cannot terminate the current function once data has been sent to the SAP System and is being processed in the SAP System. Such a situation may arise if you enter selection criteria that are not detailed enough when starting a process for finding objects in a class in the SAP classification system, and the class contains a large number of objects. You cannot terminate the function while the system is processing your selection criteria.
- If you terminate a CAD interface function with the *BREAK* function, the SAP-CAD interface program ensures that the current function is terminated correctly. For example, any lock entries created are reset. The SAP-CAD interface program is initialized so that you can continue with the next CAD interface function.

The *BREAK* function is also simple enough not to require a sub-process code. Once the current CAD interface function has been terminated, the SAP-CAD interface program sends the appropriate message to the external application program.

Communication flow for the BREAK function

SP code	Direction	Data transferred	Length
	→	None	
	←	Message	variable

Communication example:

```
BREAK        000000
```

```
BREAK  00070000000 SCH003 The 'find objects via class' function was
terminated
```

This function is called from the external application program as follows:

```
hd = SapBreak(msg).
```

SYSRQ - System Functions

This function includes several system functions that you can use to synchronize the external application program and the SAP System.

- You may need to synchronize the systems from time to time to ensure that the transferred data strings whose structure is defined in customizing are interpreted correctly.
- You can use system functions to display certain system parameters of the SAP System from the external application program.
- You can display technical information from the SAP Repository on the data fields transferred via the CAD interface functions.

Sub-functions

Function *SYSRQ* currently consists of three sub-functions that are distinguished from each other by different sub-process codes.

- Displaying System Parameters [Seite 98]
- Displaying Configuration Tables TCIM, TCIU, and TCID [Seite 100]
- Displaying Technical Field Information for a Function [Seite 102]

Displaying System Parameters

This function is identified by sub-process code *PA*.

The function is used to transfer certain system parameters of the SAP System to the external application program. At the time of writing, the following system parameters are sent to the external application program:

- **Release of the SAP System**
- **Matchcode identifier**

  This character is used in the SAP System to identify an entry as a matchcode entry. The character is defined globally in the SAP System and is usually the "=" sign.
- **Matchcode separator**

  This character is used in the SAP System to separate different fields in a matchcode entry from each other. The character is defined globally in the SAP System and is usually the "." character.
- **Separator for data strings**

  This character is used in all data strings transferred via the CAD interface whose structure cannot be defined in customizing. The character is used to separate the data fields that are transferred with variable length from each other in the data string. The separator character itself is defined in customizing functions for the CAD interface. Please note that you should only use characters that do not usually occur in ordinary text as separators.
- **Decimal point**

  The decimal point used in the SAP System is defined in the user master record of each user. You can define either “,” or “.” as a decimal point. The decimal point defined for the user that you use to logon to the SAP System is transferred. If you transfer numbers with decimal places via a CAD interface function, always use the decimal point transferred here. The other character is used as a separator for thousands.
- **Date format**

  The date format used in the SAP System is defined in the user master record of each user. You can define "DD.MM.YYYY", "MM/DD/YYYY", "MM-DD-YYYY", "YYYY.MM.DD", and "YYYY/MM/DD". The date format defined for the user that you use to logon to the SAP System is transferred. If you transfer dates via a CAD interface function, always convert them to the format defined here. Other formats trigger an error message or lead to incorrect interpretation of the date in the SAP System. You can leave out the masking characters ".", "/", and "-" and the first two figures of the year. If you enter the year as "YY", the SAP System converts any value for “YY” that is greater than “49” to "19YY", and any other value for "YY" to "20YY".

Communication flow for function SYSRQ PA

SP code	Direction	Data transferred	Length
PA	→	None	
PA	←	SAP release	4
		Matchcode identifier	1
		Matchcode separator	1

		Separator in data strings	1
		Decimal point	1
		Date format	10

Communication example:

SYSRQPA 000000

SYSRQPA00000000018 30A =.@,DD.MM.YYYY

Displaying Configuration Tables TCIM, TCIU, and TCID

This function is identified by sub-process code *TB*.

It allows you to transfer the customizing settings for the CAD interface that are defined in tables TCIM, TCIU, and TCID to the external application program. These tables contain information on the structure of the data strings transferred in the individual CAD interface functions. However, this does not include the functions whose structure is predefined by SAP. Furthermore, only the table extracts that are relevant to establishing the connection to the specific external application program are sent. The way this works is that only the entries in table TCIM that belong to the CAD identifier used in function *CONNC* are included.

The table entries are transferred to the external application program in three consecutive steps. This means that, after the function call from the external application program, three consecutive data strings are sent back to the external application program:

1. The first data string contains the relevant information from table TCIM.
2. The second data string contains entries from table TCIU.
3. The third data string contains entries from table TCID.

Only the entries in tables TCIU and TCID that belong to functions supported for the current external system identifier are sent. This is defined in table TCIM

Communication flow for function SYSRQ TB

SP code	Direction	Data transferred	Length
TB	→	None	
TM	←	Process code &) Sub-process code &) Send direction &) Set name &) This sequence is transferred for each entry in table TCIM that belongs to the current external system identifier.	variable variable variable variable
TU	←	Set name &) Local field name &) Field length &) This sequence is repeated for each relevant entry in table TCIU (according to table TCIM and current external system).	variable variable variable
TD	←	Set name &) Local field name &) Field length &) This sequence is repeated for each relevant entry in table TCID (according to table TCIM and current external system).	variable variable variable

Communication example:

SYSRQTB 000000

SYSRQTM00000000912XBOMCN@MK@D@BORQDAT@BOMCN@MK@U@BOCNDAT@...

SYSRQTU00000002219XBOCNDAT@MATERIAL@018@BOCNDAT@PLANT@004@...

SYSRQTD00000001518 BORQDAT@VALID_FROM@010@BORQDAT@VALID_TO@010@.

This function is called from the external application program as follows:

hd = SapSysrq(scode, sndstr, rcvstr, msg).

Displaying Technical Field Information for a Function

This function is identified by sub-process code *DO*.

The function allows you to access the technical attributes of the data fields transferred in a CAD interface function. This information is taken from the SAP Repository. This function can only be executed for a CAD interface function if the structure of the data string was defined via customizing.

To display the technical information on data fields transferred in a CAD interface function, enter the following data for the function call of function *SYSRQ DO*:

- Process code
- Sub-process code
- Send direction of the communication step

Communication flow for function SYSRQ DO

SP code	Direction	Data transferred	Length
DO	→	Process code &) Sub-process code &) Send direction &)	variable variable variable
DO	←	Local field name &) SAP field name (keyword) &) Data type &) Field length &) Number of decimal places &) “Check table” indicator &) Name of check table &) Template &) This sequence is repeated for each entry in table TCIU or TCID that belongs to the field set used in the function entered.	variable variable variable variable variable variable variable variable

Communication example:

You want to display the technical field information for the data fields sent from the external system to the SAP System when you create a new material master record (function *MATRQ MR*) (send direction “U“).

SYSRQDO 000011 MATCR@MR@U@

SYSRQDO00000001122 MATERIAL@Material number@CHAR@018@00@@@@...

HLPRQ - Help Functions

This function makes the online help functions for screen fields that are supported in the SAP System accessible in the external application program.

There are two types of online help:

- You can call online documentation on an SAP screen field.
- You can display possible entries for an SAP screen field. These are defined in a check table or as domain values.

You can only display online field documentation and possible entries for characteristics of classes and for the data fields that can be defined in customizing functions. These help functions are not supported for functions where the data fields transferred are predefined.

You can call these help functions at any time. You can call this function at any time, even between two communication steps of another CAD interface function.

Sub-functions

Function *HLPRQ* currently comprises four sub-functions that are distinguished by different sub-process codes:

- Displaying ONLINE Field Documentation (F1 Help) [Seite 104]

  This help function allows you to access the online help functions for screen fields that are supported in the SAP System directly from the window in the external application program. In the customizing functions for the CAD interface, you use the local field names to define the correlation between the screen fields of the external application program and the SAP screen fields (or the data fields of the SAP Repository). This means that when you display online help on a field, you need to enter the local field name, the process code, the sub-process code, and the send direction that define the communication step in which the data field is transferred.
- Displaying Possible Entries (F4 Help) [Seite 106]
- Displaying Allowed Values for Several Characteristics [Seite 108]
- Displaying the Long Text for an Error Message [Seite 109]

Displaying ONLINE Field Documentation (F1 Help)

This function is identified by sub-process code *SD.*

It is used to make the field documentation defined in the SAP System for a screen field of the external application program available in the external application program. The documentation received from the SAP System is already formatted correctly and can be displayed directly line for line in the external application program, without prior processing. Consecutive lines of text are separated by a separator character.

Communication flow for function HLPRQ SD

Table Caption

SP code	Direction	Data transferred	Length
SD	→	Process code &) Sub-process code &) Send direction &) Local field name, if standard data field &) Characteristic name, if characteristic &)	variable variable variable variable variable
SD	←	First line &) Second line &) . . . Last line &)	variable variable . . . variable

If you want to display the field documentation for a normal screen field, you must enter the corresponding local field name using the CAD interface. If you want to display a characteristic for a class, you do not need to transfer the local field name. You should use the language-independent characteristic name for the characteristic instead. This characteristic name is sent from the SAP System to the external system in a classification function. This name only exists once within the SAP System and uniquely identifies the characteristic.

Communication example:

You want to display the field documentation for the *Document number* field that is sent from the external application program to the SAP System when you create a new document info record. The local field name defined for the document number in the customizing function for the CAD interface is, say, DOC_NUMBER. The communicaton step in which this field is transferred is *DOCCR MR*. In other words, the process code is *DOCCR,* the sub-process code is *MR,* and the send direction is “U“ (from the external system to the SAP System).

HLPRQSD 000023 DOCCR@MR@U@DOC_NUMBER@@

HLPRQSD00000001074 Feld:Dokument@@ Dokumentnummer@@Definition@@ Nummer, die ein Dokument als Hauptidentifiziert.@@Vorgehen:@@ Beachten Sie bitte beim Anlegen:@ Sie haben zwei Möglichkeiten,...@

Communication example:

You want to display the field documentation for a characteristic with characteristic name D_BDREHZAHL.

HLPRQSD 000016 @@@@D_BDREHZAHL@

HLPRQSD00000000213 Dieses Merkmal enthält die maximal zulässigen Betriebsdrehzahlen der@ Zahnradpumpe. Sie beträgt im Falle einer Pumpe mit Ausgleichskupplung@ 2500 U/min und für eine Kupplung mit starrer Kupplung 1500 U/min.@

Displaying Possible Entries (F4 Help)

This function is identified by sub-process code *SE*.

The function is used to make the possible entries defined in the SAP System for a screen field of the external application program available in the external application program.

Communication flow for function HLPRQ SE

SP code	Direction	Data transferred	Length
SE	→	Process code &)	variable
		Sub-process code &)	variable
		Send direction &)	variable
		Local field name, if standard data field &)	variable
		Characteristic name, if characteristic &)	variable
ST	←	Heading of first column &)	variable
		Width of first column &)	variable
		Selection indicator of first column &)	variable
		.	.
		.	.
		.	.
		Heading of last column &)	variable
		Width of last column &)	variable
		Selection indicator of last column &)	variable
SE	←	First value for first column &)	variable
		First value for second column &)	variable
		.	.
		.	.
		.	.
		First value for last column &)	variable
		.	.
		.	.
		.	.
		Last value for first column &)	variable
		Last value for second column &)	variable
		.	.
		.	.
		.	.
		Last value for last column &)	variable

The structure of the data string to be sent to the SAP System when the function is called is the same as for the data string in function *HLPRQ SD*.

The table above shows that the SAP System sends two data strings to the external application program after the function call.

- The first data string contains the heading line that belongs to the check table. The number of headings corresponds directly to the number of columns of the current check table displayed. The number of columns is variable and can be up to a maximum of six. In addition to the heading of each column, you see the width for display in the external program. This is already defined such that there will be just enough space for the widest entry in this column. An indicator “X” appears in a column that contains selectable values. This indicator is not set in any other columns.

- The second data string contains the values in the check table. These values are assigned consecutively to the individual columns of the table.

Communication example:

You want to display possible entries for the *Equipment category* that is sent from the external application program to the SAP System when you create a new equipment master record. The local field name defined for the equipment category in the customizing function for the CAD interface is, say, EQUI_CATEG. The communicaton step in which this field is transferred is *EQICR MR*. In other words, the process code is *EQICR,* the sub-process code is *MR,* and the send direction is “U“ (from the external system to the SAP System).

HLPRQSE 000023 EQICR@MR@U@EQUI_CATEG@@

HLPRQST00000000035XEquipmenttyp@12@X@Beschreibung@30@@

HLPRQSE00000000150 D@DV Equipment@M@Maschinen@P@Fertigungshilfsmittel@Q@Prüf- und Meßmittel@R@Rechenzentrumsequipment@S@Kundenequipment@T@Telekommunikation@ Z@TN Equityp@

Communication example:

You want to display the possible entries for a characteristic with characteristic name D_BDREHZAHL.

HLPRQSE 000016 @@@@D_BDREHZAHL@

HLPRQST00000000022XBetriebsdrehzahl@40@X@

HLPRQSE00000000022 1500 U/MIN@2500 U/MIN@

Displaying Allowed Values for Several Characteristics

This function is identified by sub-process code *SC*.

It can only be used for characteristics. If you want to display the possible entries for several characteristics of a class in a classification function, you can display all the values for several characteristics with one function call. This improves performance relative to individual function calls per characteristic.

Communication flow for function HLPRQ SC

SP code	Direction	Data transferred	Length
SC	→	Characteristic name of first characteristic &) . . . Characteristic name of last characteristic &)	variable . . . variable
SC	←	Characteristic name of first characteristic &) First value of first characteristic &) . . . Characteristic name of first characteristic &) Last value of first characteristic &) . . . Characteristic name of last characteristic &) First value of last characteristic &) . . . Characteristic name of last characteristic &) Last value of last characteristic &)	variable variable . . . variable variable . . . variable variable . . . variable variable

Communication example:

You want to display the possible entries for all characteristics with characteristic names D_OELDRUCK, D_BDREHZAHL, and D_KUPPLUNG.

HLPRQSC 000034 D_OELDRUCK@D_BDREHZAHL@D_KUPPLU@

NG

HLPRQSC00000000108 D_OELDRUCK@50 BAR@D_OELDRUCK@70

BAR@D_BDREHZAHL@1500 U/MIN@D_BDREHZAHL@2500 U/MIN@

D_KUPPLUNG@S@D_KUPPLUNG@N@

Displaying the Long Text for an Error Message

This function is identified by sub-process code *MT*.

The function is used to display the long text of an error message received from the SAP System. This text contains further information on the error that has occurred. The data received from the SAP System when you display the long text contains the text in a formatted form. This text can be displayed directly line by line in the external system. Consecutive lines are separated by a separator character.

To display the long text of an error message, you need to send the identifier of the message to the SAP System with the function call. This identifier comprises a two-character message ID and a three-character message number. You will find both of these in any message that is sent to the external application program in case of error.

Each message received from the SAP System comprises a six-byte header section followed by the text section:

- The first byte of the header section contains the message type.
- The next two bytes contain the message ID.
- The last three bytes contain the message number.

  Example: *EM3307 Material already maintained for this operation*

  Message type: E; Message ID: M3; Message number: 307.

Communication flow for function HLPRQ MT

SP code	Direction	Data transferred	Length
MT	→	Message ID &) Message number &)	variable variable
MT	←	First line of long text &) Second line of long text &) . . . Last line of long text &)	variable variable . . . variable

Communication example:

You want to display the long text of error message EM3307, shown above.

HLPRQMT 000007 M3@307@

HLPRQMT00000000262 Diagnose@ Der Materialstammsatz ist f

ür die ausgewählte Sicht oder@ Organisationsebene bereits an gelegt.@ Vorgehen@ Prüfen Sie Ihre Eingaben

This function is called from the external application program as follows:

hd = SapHlprq(scode, sndstr, rcvstr, msg).

Functions for Material Master Records

The CAD interface supports the following functions for processing material master records:

MATCR - Creating a Material Master Record [Seite 111]

MATCH - Changing a Material Master Record [Seite 122]

MATRQ - Displaying a Material Master Record [Seite 129]

RSRVE - Reserving a Material Number [Seite 137]

MATCR - Creating a Material Master Record

The CAD interface enables you to create material master records in the SAP System from the external application program.

You can maintain all views on a material master record except long texts (sales text and purchase order text). As of release 3.0, you can also maintain the *Classification* view via the CAD interface.

There are two CAD interface functions with different sub-process codes for creating a new material master record. This is because there are different options for number assignment.

- Function *MATCR MR* is used to create a material master record in the SAP System via the CAD interface with external or internal number assignment.

  However, especially in the case of an external PDM system with its own database, this can give rise to problems. If an external PDM system is used in your design department, product data produced during the design process, such as data on parts and bills of material, are managed in the PDM system first. The data is not transferred to the SAP System until it is released. However, unless you ensure first that a material can be created in the SAP System with the number assigned in the PDM system, such a procedure can mean that the material number must be changed. To solve this problem, the CAD interface supports a function for reserving material numbers in the SAP System.

- To create a material master record record in the SAP System via the CAD interface with a previously reserved material number, use function *MATCR MA*.

When you call the function, you send the material master data to the SAP System according to the structure defined in table TCIU.

Further processing depends on whether you want to classify the material or not.

The following sections contain more detailed information:

- Creating a Material Master Record Without Classification [Seite 112]
- Creating a Material Master Record With Classification [Seite 113]
- Setting up Strings for Characteristics and Values [Seite 115]

Creating a Material Master Record Without Classification

The table below shows the communication flow for materials that are **not** to be classified. Once the material data is sent to the SAP System, the material master record is created and the message displayed by the appropriate transaction is returned to the external system.

Communication flow for function MATCR MR without classification

SP code	Direction	Data transferred	Length
MR	→	Material master data according to table TCIU	variable
MR	←	Function message	variable

Creating a Material Master Record With Classification

If you want to classify the material the communication flow is more extensive, because the characteristic information also has to be transferred. When you send the general material master data, you must send the class type and class required at the same time.

You can also specify the scope to be used. The scope acts as a filter for selecting the relevant characteristics of the class. The scope selects according to whether characteristics are required or optional, whether they have values assigned or not, and whether they are inherited. In a further communication step, the SAP interface program then sends the characteristics selected by the scope defined to the external application program, together with any default values.

In the external system, you assign values to the characteristics and transfer them to the SAP System. You can use the help functions *HLPRQ SC* or *HLPRQ SE* to display the allowed characteristic values. The material master record is created in the SAP System with its classification data.

Communication flow for function MATCR MR with classification

SP code	Direction	Data transferred	Length
MR	→	Material master data according to table TCIU including class type, class, and scope	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Assigned Characteristic Values	variable
MR	←	Function message	variable

After communication step *MATCR CR* (SAP→CAD), you can either assign values to the characteristics or continue with a changed combination of class type, class, and scope.

When you first create a material, you can only classify the material in one class. You cannot classify the document in several classes at once in this way. However, you can do a multiple classification of a material using the function ‘Change material master record with classification’ *MATCH MC*.

Creating a Material Master Record With Classification

See also:

Setting up Strings for Characteristics and Values [Seite 115]

Setting up Strings for Characteristics and Values

The following communication steps are involved in setting up characteristic and value strings:

- Characteristics Data for a Class [Seite 116]

  Setting up the data string that contains technical information on the characteristics
- Values for Characteristics [Seite 119]

  Setting up the data string that contains the default values of the characteristics
- Assigned Characteristic Values [Seite 120]

  Setting up the data string that contains the values assigned to the characteristics

Characteristics Data for a Class

The structure of the data string sent to the external application program in communication step *MATCR RF* (SAP→CAD) is shown in the table below. This table describes the structure of the data string containing **technical information on the characteristics**.

Setting up the data string that contains technical information on the characteristics

SP code	Direction	Data transferred	Length

RF	←	Class type &)	variable
		Class &)	variable
		Class description &)	variable
		Status text &)	variable
		Indicator: standard class &)	variable
		Name of first characteristic &)	variable
		Description of first characteristic &)	variable
		Indicator: required characteristic &)	variable
		Indicator: inherited characteristic &)	variable
		Indicator: characteristic with assigned values &)	variable
		Data type of characteristic &)	variable
		Indicator: single value &)	variable
		Indicator: interval values allowed &)	variable
		Indicator: additional values &)	variable
		Number of characters &)	variable
		Number of decimal places &)	variable
		Indicator: negative values allowed &)	variable
		Exponent in screen display &)	variable
		Exponent display &)	variable
		Template &)	variable
		Indicator: case sensitive &)	variable
		External unit of measure (6-character) &)	variable
		DIN code letter &)	variable
		Document type &)	variable
		Document number &)	variable
		Document part &)	variable
		Document version &)	variable
		View indicator 0 &)	variable
		View indicator 1 &)	variable
		View indicator 2 &)	variable
		View indicator 3 &)	variable
		View indicator 4 &)	variable
		View indicator 5 &)	variable
		View indicator 6 &)	variable
		View indicator 7 &)	variable
		View indicator 8 &)	variable
		View indicator 9 &)	variable
		.	.
		.	.
		.	.
		Name of last characteristic &)	variable
		Description of last characteristic &)	variable
		Indicator: required characteristic &)	variable
		Indicator: inherited characteristic &)	variable
		Indicator: characteristic with assigned values &)	variable
		Data type of characteristic &)	variable
		Indicator: single value &)	variable
		Indicator: interval values allowed &)	variable
		Indicator: additional values &)	variable
		Number of characters &)	variable
		Number of decimal places &)	variable
		Indicator: negative values allowed &)	variable
		Exponent in screen display &)	variable
		Exponent display &)	variable
		Template &)	variable
		Indicator: case sensitive &)	variable
		External unit of measure (6-character) &)	variable
		DIN code letter &)	variable
		Document type &)	variable
		Document number &)	variable
		Document part &)	variable

Values for Characteristics

The structure of the data string sent to the external application program in communication step *MATCR CR* (SAP→CAD) is described in the table below. This table describes the structure of the data string that contains **default values for characteristics**.

Setting up the data string that contains the default values of the characteristics

SP code	Direction	Data transferred	Length
CR	←	Name of first characteristic &)	variable
		(Default) value for the first characteristic &)	variable
		.	.
		.	.
		.	.
		Name of last characteristic &)	variable
		(Default) value for the last characteristic &)	variable

Assigned Characteristic Values

In communication step *MATCR CR* (CAD→SAP), the **values assigned to characteristics** in the external system are sent to the SAP System. The structure of the data string is described in the table below. Please note that a characteristic may occur more than once in this list. This is the case if multiple values are to be assigned to the characteristic.

Setting up the data string that contains the values assigned to the characteristics

SP code	Direction	Data transferred	Length
CR	→	Name of first characteristic &) Value for the first characteristic &) . . . Name of last characteristic &) Value for the last characteristic &)	variable variable . . . variable variable

If no data on assigned characteristic values is sent to the SAP System in this communication step - in other words, if the data length of the send string is ‘0’ - the document info record is just allocated to the class. No characteristic values are assigned in this case.

Number assignment

In the customizing functions for the CAD interface, you can define whether document numbers can be assigned externally only, internally only, or both.

The communication procedures apply as for the function *MATCR MA*. However, the definition of this function does not allow internal number assignment. Furthermore, the material number you enter must be reserved beforehand using function *RSRVE MT*.

Communication example:

You want to create material CAD-MOTOR with views for ‘Engineering/design’ and ‘Classification’.

MATCRMR 000199 CAD-MOTOR XX HALBMSportmotor

ST 40 X 40 X 30 001MP_MOTOR XXXXXX

MATCRRF00000000281X001@MP_MOTOR@Motor@@@MP_LEISTUNG@Leistung@X@@@NUM@X@@@3@0@@0@0@___@X@KW@@@@@@@@@@@@@@@@@MP_HUBRAUM@Hubraum@@@@NUM@X@@@2@1@@0@0@_,_@X@l@@@@@@@@@@@@@@@@MP_GEWICHT@Gewicht@@@@NUM@X@@@5@1@@0@0@_.___,_@X@kg@@@@@@@@@@@@@@@@MP_TREIBSTOFF@Treibstoff@@@@CHAR@X@@@6@0@@0@0@@X@@@@@@@@@@@@@@@@@

MATCRCR00000000018 MP_LEISTUNG@90 KW@HLPRQSC 000048 MP_LEISTUNG@MP_HUBRAUM@MP_GEWICHT@MP_TREIBSTOFF@

HLPRQSC00000000042 MP_LEISTUNG@10 - 300 KW@MP_LEISTUNG@90KW@

MATCRCR 000019 MP_LEISTUNG@800 KW@

MATCRCR04041000011 EC1801 Wert ‘800 KW’ ist nicht vorhandenMP_ LEISTUNGMATCRCR 000018 MP_LEISTUNG@20 KW@

MATCRMR00043000018 SM3800 Das Material CAD-MOTOR wird angelegtCAD-MOTOR

This function is called from the external application program as follows:

hd = SapMatcr(scode, sndstr, rcvstr, msg).

MATCH - Changing a Material Master Record

When you change material master records, you can maintain the same views that you can maintain when you create a new material.

The following functions are supported:

- Changing a Material Master Record Without Classification [Seite 123]
- Changing a Material Master Record With Classification [Seite 124]
  - List of All Classes [Seite 126]
  - MATCH MC: Creating an Allocation to an Additional Class [Seite 127]
  - MATCH MC: Deleting an Existing Allocation to a Class [Seite 128]

Changing a Material Master Record Without Classification

This function is identified by sub-process code *MR*.

The function allows you to change data in the material master record, with the exception of data in the classification view. The communication flow of function *MATCH MR* is described in the table below.

Communication flow for function MATCH MR

SP code	Direction	Data transferred	Length
MR	→	Material master key data as defined in table TCIU	variable
MD	←	Material master data in SAP System as defined in table TCID	variable
MD	→	Changed material master data as defined in table TCIU	variable
MR	←	Function message	variable

When you call the function, the data that identifies the material master record is sent to the SAP System. Along with the material number, this data forms the indicators for the views that are to be changed and the key fields of these views, such as *Plant*, *Storage location* and so on. Then the SAP interface program sends the available material data (according to the settings in Customizing) to the external application program. At the same time, a lock entry is set in the SAP System for the material to be changed to prevent concurrent access by other users.

The changes made to the material master data in the external system are returned to the SAP System in a further communication step. The SAP interface program then ensures that the changes are saved in the SAP System.

Communication example:

You want to change the ‘Engineering/design’ view of material CAD-MOTOR.

```
MATCHMR      000019 CAD-MOTOR          X

MATCHMD00000000099 Sportmotor                                HALBMST
0,000                0,000

MATCHMD      000096
200,43            kg 195,46            A0

MATCHMR00043000000 SM3801 Material CAD-MOTOR changed
```

Changing A Material Master Record With Classification

This function is identified by sub-process code *MC*. It allows you to change data in the material master record as in function *MATCH MR.* However, you can also maintain the classification view. You have the following options:

- changing an existing allocation and/or existing characteristic values
- creating an allocation to a new class
- deleting the allocation to a class

You can only create, change, or delete the allocation to one class in one function call. You cannot process several class allocations in one function.

Please note that classification is connected to engineering change management. For this reason, all changes to classification data can be made historically with either a date or a change number.

The communication flow for function *MATCH MC* is the same as for function *MATCH MR* up to communication step *MATCH MD* (SAP→CAD). When the changed material master data is sent to the SAP System in communication step *MATCH MD* (CAD→SAP), you can also send values for the class type and class whose allocation you want to change.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes to which the material is allocated is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class. However, if a standard class is defined for the material, or if the material is only allocated to one class, the communication step for transferring a list of classes is no longer necessary. In this case, the information on characteristics and assigned values is sent directly to the external program.

The communication flow of function *MATCH MC* when changing an existing allocation is described in the table below.

Communication flow for function MATCH MC for changing an existing allocation

SP code	Direction	Data transferred	Length
MC	→	Material master key data as defined in table TCIU	variable
MD	←	Material master data in SAP System as defined in table TCID	variable
MD	→	Changed material master data as defined in table TCIU including class type, class, and scope	variable
RC	←	List of all classes to which the material is allocated	variable

RF	→	Class type &)	variable
		Class &)	variable
		Indicator: required characteristics &)	variable
		Indicator: optional characteristics &)	variable
		Indicator: characteristics with assigned values &)	variable
		Indicator: characteristics without assigned values &)	variable
		Indicator: inherited characteristics &)	variable
		Indicator: standard class &)	variable
RF	←	Characteristics Data for a Class	variable
CR	←	Assigned characteristic values	variable
CR	→	Changed assigned characteristic values	variable
MC	←	Function message	variable

See also:

List of All Classes [Seite 126]

MATCH MC: Creating an Allocation to an Additional Class [Seite 127]

MATCH MC: Deleting an Existing Allocation to a Class [Seite 128]

List of All Classes

After communication step *MATCH CR* (SAP→CAD), you can either assign characteristic values or continue with a different combination of class type, class, and scope.

Communication step *MATCH RC* (SAP→CAD) only applies if the restrictions sent in communication step *MATCH MD* (CAD→SAP) produced a list of several classes in which the material is classified.

The table below describes the structure of the data string that contains the **List of classes in which the material is classified**.

Structure of the data string that contains the list of classes in which the material is classified

SP code	Direction	Data transferred	Length
RC	←	Class type of first class &)	variable
		First class &)	variable
		Description of first class &)	variable
		Status text for allocation to class &)	variable
		Indicator: standard class &)	variable
		.	.
		.	.
		.	.
		Class type of last class &)	variable
		Last class &)	variable
		Description of last class &)	variable
		Status text for allocation to class &)	variable
		Indicator: standard class &)	variable

MATCH MC: Creating an Allocation to an Additional Class

If your change to classification data is an allocation to an additional class, the communication flow is as follows:

Communication flow for function MATCH MC for creating an allocation to an additional class

SP code	Direction	Data transferred	Length
MC	→	Material master key data as defined in table TCIU	variable
MD	←	Material master data in SAP System as defined in table TCID	variable
MD	→	Changed material master data as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], that the material is allocated to.	variable
NC	→	New class type &) New class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119]	variable
RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

MATCH MC: Deleting an Existing Allocation to a Class

If you want to delete an existing allocation to a class, the communication flow is as follows:

Communication flow for function MATCH MC for deleting an existing allocation

SP code	Direction	Data transferred	Length
MC	→	Material master key data as defined in table TCIU	variable
MD	←	Material master data in SAP System as defined in table TCID	variable
MD	→	Changed material master data as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126]	variable
DC	→	Class type to be deleted &) Class to be deleted &)	variable variable
MC	←	Function message	variable

Communication example:

You want to change the ‘Classification’ view of CAD-MOTOR.

MATCHMC 000020 CAD-MOTOR X

MATCHMD00000000099 Sportmotor HALBMST
0,000 0,000

MATCHMD 000000

MATCHRF00000000283X001@MP_MOTOR@Motor@@@MP_Leistung@Leistung@X@@X@
NUM@X@@@3@0@@0@0@___@X@KW@@@@@@@@@@@@@@@@MP_HUBRAU
M@Hubraum@@@@NUM@X@@@2@1@@0@0@_,_@X@1@@@@@@@@@@@@@@
@@@MP_GEWICHT@Gewicht@@@@NUM@X@@@5@1@@0@0@_.___,_@X@kg@@@
@@@@@@@@@@@@@@MP_TREIBSTOFF@Treibstoff@@@X@CHAR@X@@@6@0@
@0@0@@X@@@@@@@@@@@@@@@@@@

MATCHCR00000000039 MP_LEISTUNG@40 KW@MP_TREIBSTOFF@BENZIN@

MATCHCR00000000039 MP_LEISTUNG@30 KW@MP_TREIBSTOFF@BENZIN@

MATCHMR00052000000 SCH100 Das Material ‘CAD-MOTOR ‘ wurde klassifiziert

This function is called from the external application program as follows:

hd = SapMatch(scode, sndstr, rcvstr, msg).

MATRQ - Displaying a Material Master Record

The following functions for displaying material master data are supported:

- Displaying a Material Master Record Without Classification [Seite 130]
- Displaying a Material Master Record With Classification [Seite 131]
- Finding a Material Master Record via Matchcode [Seite 133]
- Displaying Material Master Record Data for Several Materials (Mass Access) [Seite 136]

Displaying a Material Master Record Without Classification

This function is identified by sub-process code *MR*.

The function allows you to display data in the material master record, with the exception of data in the classification view. The communication flow of function *MATRQ MR* is described in the table below.

Communication flow for function MATRQ MR

SP code	Direction	Data transferred	Length
MR	→	Material master key data as defined in table TCIU	variable
MR	←	Material master data in SAP System as defined in table TCID	variable

When you call the function, the data that identifies the material master record is sent to the SAP System. Along with the material number, this data forms the indicators for the views that are to be changed and the key fields of these views, such as *Plant*, *Storage location* and so on. Then the SAP interface program sends the available material data (according to the settings in Customizing) to the external application program.

Displaying a Material Master Record With Classification

This function is identified by sub-process code *MC*.

The function allows you to display data in the material master record as in function *MATRQ MR.* You can also display the classification view. The communication flow of function *MATRQ MC* is described in the table below.

Communication flow for function MATRQ MC

SP code	Direction	Data transferred	Length
MC	→	Material master key data as defined in table TCIU including class type, class, and scope	variable
MC	←	Material master data in SAP System as defined in table TCID	variable
RC	←	List of All Classes [Seite 126], that the material is allocated to.	variable
RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119]	variable
RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119]	variable

Displaying a Material Master Record With Classification

When you call function *MATRQ MC,* you can enter values for the *Class type* and *Class* whose allocations you want to display immediately.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes to which the material is allocated is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class.

In the following situations, the information on characteristics and assigned values is sent directly to the external program:

  - The restrictions in force when you call the function apply to only one class
  - The material is only allocated to one class.

- You can also enter a date or a change number. Since classification is made with history, the status on a specific date is displayed. If you do not enter a date, the current date is taken as a default.

  - If the restrictions on classification that you send with the function call are such that **several** classes are selected, this list of classes is first sent to the external application program in communication step *MATRQ RC* (SAP→CAD).

    In the external system, you select the class you are interested in and return the selected class to the SAP System in communication step *MATRQ RC* (CAD→SAP).

    You then see the characteristic and characteristic values for the class and scope you require.

    You can repeat communication step *MATRQ RC* (CAD→SAP) as many times as you like. This allows you to display the classification data of the material in each class in the list and to vary the scope. Since there is no obvious point at which to end function *MATRQ MC,* please note that you must use function *BREAK* to terminate this function.

  - If only **one** class is selected or if the material is classified in exactly one class,
  - you can only vary the scope in communication step MATRQ RC (CAD→SAP).

In function *MATRQ RC* (CAD→SAP), you can only display the classification data that belongs to classes selected when you called the function.

Classification data for classes that were not selected cannot be displayed. This means that if the material is classified in classes of two different class types, and you enter a class type when you call function *MATRQ MC* (CAD→SAP), you do not see the allocations to classes of the second class type. You cannot display these allocations in function *MATRQ RC* (CAD→SAP) by entering the other class type, either. You need to call function *MATRQ MC* again.

Finding a Material Master Record via Matchcode

This function is identified by sub-process code *RL*.

It allows you to find a material in the SAP System if you do not know the material number, but know other master data, such as the description or the basic material. You can also find materials using classification data. This function is described in the section on the classification system.

The matchcode functionality allows you to define a matchcode ID such that it starts the function for finding objects via classes or a report, instead of the standard matchcode search function. These matchcode IDs cannot be processed using the CAD interface. If you select one of these matchcode IDs, you see an error message telling you that the matchcode ID cannot be used.

When you use a matchcode, either:

- You do not know the field structure of the selected matchcode ID, or
- You know the field structure of the selected matchcode ID.

The communication flow of function *MATRQ RL* is described in the tables below.

Communication flow for function MATRQ RL if you do not know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &)	variable
		Matchcode sub-ID &)	variable
		Description of matchcode ID &)	variable
		.	.
		.	.
		.	.
		Last matchcode ID &)	variable
		Matchcode sub-ID &)	variable
		Description of matchcode ID &)	variable
ID	→	Selected matchcode ID &)	variable
		Matchcode sub-ID &)	variable

Finding a Material Master Record via Matchcode

RF	←	First matchcode field &) Length of first matchcode field &) . . . Last matchcode field &) Length of last matchcode field &)	variable variable . . . variable variable
RF	→	Matchcode selection string	variable
CL	←	Material master data in SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Material master data in SAP System as defined in table TCID	variable

Communication flow for function MATRQ RL if you know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Matchcode selection string	variable
CL	←	Material master data in SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Material master data in SAP System as defined in table TCID	variable

The matchcode sub-ID is still included in the transferred data strings to ensure compatibility with R/2. In data strings sent from the R/3 System to the external application program, the matchcode sub-ID is SPACE. In data strings sent from the external application program to the SAP System, you can either send the additional separator character or not.

After you call function *MATRQ RL,* first the matchcode IDs defined in the SAP material master are sent to the external application program.

For further processing, there is a difference depending on whether you know the field structure of the selected matchcode or not.

- If you do **not** know the field structure, you send the selected matchcode ID to the SAP System in the next communication step in order to get the field structure.
- However, if you know the field structure, you can use the next communication step to send the matchcode selection string. To do this, you use the matchcode identifier (usually the '0' character). The matchcode identifier is transferred to the external application program using system function *SYSRQ PA*.

When you use matchcodes to find material master records, please note that the master record of each material found by the matchcode search is read in order to procure the data that is not stored in matchcode records. However, this may lead to runtime problems if the matchcode selection criteria are not specific enough and therefore a large number of material master records is selected.

For this reason, control table TCIS contains parameter TCIS-MCMAX, which defines the maximum number of matchcode records, after which communication with the external application program is triggered. If such a communication is necessary, you use sub-process code *CL*. If you call function *MATRQ CL,* you can then request the following matchcode records.

As of release 3.0, all matchcode selection functions were changed such that functions *MATRQ ID* and *MATRQ RF* can be called in any communication step. This has the advantage that you do not need to start again by calling function *MATRQ RL* if your matchcode search is unsuccessful. Instead, you can enter new values in the matchcode fields directly if you want to use the current matchcode ID, or switch to another matchcode ID. However, this means that the matchcode selection functions no longer come to a natural end. As of release 3.0, you terminate these functions using the *BREAK* function

Displaying Material Master Record Data for Several Materials (Mass Access)

This function is identified by sub-process code *SM*. It allows you to display material master data for a list of materials. This is known as mass access. However, you can only display general material master data from tables MARA and MAKT using this function. You can use this function if, for example, you want to read the material descriptions of all BOM material items in one transaction.

The communication flow of function *MATRQ SM* is described in the table below.

Communication flow for function MATRQ SM

SP code	Direction	Data transferred	Length
SM	→	First material number &) . . . Last material number &)	variable . . . variable
SM	←	Material master data in SAP System as defined in table TCID	variable

When you call this function, the material numbers of all the materials you are interested in are sent to the SAP System. These material numbers are separated from each other by a separator character.

The material master data is read and sent to the external application program in a data string as defined in settings in table TCID. Please note that you do not see an error message if individual material master records do not exist in the SAP System. These materials are just ignored.

This function is called from the external application program as follows:

hd = SapMatrq(scode, sndstr, rcvstr, msg).

RSRVE - Reserving a Material Number

The CAD interface gives you the option of reserving material numbers from an internal number range in the SAP System. This is a useful function if you are working with a PDM system.

If you are using a PDM system to design a new part, a local part master generally exists before the part is released. Once the part is released, the part is transferred to the SAP System via the CAD interface. As a rule, it will not be possible to create the material master record in the SAP System with the part number you assigned in the PDM system. This means that the part number has to be changed when you transfer the part to the SAP System.

To avoid this situation, the CAD interface gives you the option of reserving a material number in the SAP System before you start. This does not mean that you create a material master record in the SAP System, but it ensures that you can use the reserved number to create a material master record in the SAP System when you release the part.

The communication flow of function *RSRVE MT* is described in the table below.

Communication flow for function RSRVE MT

SP code	Direction	Data transferred	Length
MT	→	Material type (optional)	variable
MT	←	Reserved material number	variable

When you call this function, you can send a value for the material type. The material type determines the number range from which the material number is taken.

- If you send a value for the material type, the next available number from the number range defined for this material type is reserved. You can define a default value for the material type in control table TCIS.
- If you do **not** send a value for the material type when you call the function, the default value from table TCIS is used. You can also switch the material number reservation option on or off in table TCIS.

If you want to create a material master record with a reserved material number via the CAD interface, call function *MATCR MA*. When you create the material master record, you can enter a different material type to the one you used for reserving the material number.

This function is called from the external application program as follows:

hd = SapRsrve(scode, sndstr, rcvstr, msg).

Functions for Document Management

The following functions for processing document info records are supported by the CAD interface.

DOCCR - Creating a Document Info Record [Seite 139]

DOCCH - Changing a Document Info Record [Seite 143]

DOCRQ - Displaying a Document Info Record [Seite 151]

DOCCI - Check In Functions for Original Application Files [Seite 165]

DOCCO - Check Out Functions for Original Application Files [Seite 169]

The abbreviation *document* is sometimes used in this documentation to refer to a *document info record*.

DOCCR - Creating a Document Info Record

The CAD interface gives you the option of creating document info records in the SAP System from the external application program. You can maintain all document info record data except long texts.

- Up to release 2.2, you can also create object links when you create a document info record. However, you can only link one object of each type to the document using the CAD interface.
- As of release 3.0, you can use function *DOCCR MR* to create the document info record only. The CAD interface has separate functions for maintaining object links.

When you call the function, you must send the document info record data to the SAP System in the structure defined in table TCIU. The communication flow that follows depends on whether you want to classify the document info record or not.

- Creating a Document Info Record Without Classification [Seite 140]
- Creating a Document Info Record With Classification [Seite 141]

Creating a Document Info Record Without Classification

The table below shows the communication flow for creating a document info record without classifying it. Once you have sent the data to the SAP System, the document info record is created and the message displayed by the create transaction is returned to the external system.

Communication flow for function DOCCR MR without classification

SP code	Direction	Data transferred	Length
MR	→	Document data as defined in table TCIU	variable
MR	←	Function message	variable

Creating a Document Info Record With Classification

If you want to classify the document you are creating, the communication flow is more extensive. This allows you to transfer characteristics data.

- If you want to classify the document, you must send the *class type* and *class* you require with the general document data.
- You can also specify the scope to be used. The scope acts as a filter for selecting the relevant characteristics of the class. The scope selects according to whether characteristics are required or optional, whether they have values assigned or not, and whether they are inherited.
- In the next communication step, the SAP interface program sends the characteristics as defined in the scope you entered, together with any default values, to the external application program.
- In the external system, you assign values to the characteristics. You can use the help functions *HLPRQ SC or HLPRQ SE* to display the allowed values. The assigned values are transferred to the SAP System. In the SAP System, the document info record is created with its classification data.

Communication flow for function DOCCR MR with classification

SP code	Direction	Data transferred	Length
MR	→	Document data as defined in table TCIU including class type, class, and scope	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Assigned Characteristic Values [Seite 120]	variable
MR	←	Function message	variable

After communication step *DOCCR CR* (SAP→CAD), you can either assign characteristic values or continue with a changed combination of class type, class, and scope.

Creating a Document Info Record With Classification

- When you create a new document info record, you can classify the document in no more than one class. You cannot classify the document in several classes at once in this way. However, you can do multiple classifications with the *Change document info record with classification DOCCH MC* function.
- In communication steps *DOCCR RF* (SAP→CAD) and *DOCCR CR*

  (SAP→CAD), the technical information on the characteristics and the default values of the characteristics are sent to the external application program.
- In communication step *DOCCR CR* (CAD→SAP), the characteristic values assigned in the external system are sent to the SAP System. Please note that a characteristic may occur more than once in this list. This is the case if multiple values are assigned to a characteristic.

  If no data on assigned characteristic values is sent to the SAP System in this communication step - in other words, if the data length of the send string is ‘0’ - the document info record is just allocated to the class. No characteristic values are assigned in this case.
- In the customizing functions for the CAD interface, you can define whether document numbers can be assigned externally only, internally only, or both.

This function is called from the external application program as follows:

hd = SapDoccr(scode, sndstr, rcvstr, msg).

DOCCH - Changing a Document Info Record

The following functions for changing a document info record are supported by the CAD interface.

- Changing a Document Info Record Without Classification [Seite 144]
- Changing a Document Info Record With Classification [Seite 145]
- Changing a Document Info Record (Object Links Only) [Seite 148]

Changing a Document Info Record Without Classification

This function is identified by sub-process code *MR*.

The function allows you to change all document info record data except classification data and object links. The communication flow of function *DOCCH MR* is described in the table below.

Communication flow for function DOCCH MR

SP code	Direction	Data transferred	Length
MR	→	Document key data as defined in table TCIU	variable
MD	←	Document data in the SAP System as defined in table TCID	variable
MD	→	Changed document data as defined in table TCIU	variable
MR	←	Function message	variable

When you call the function, the data that identifies the document info record is sent to the SAP System. This data comprises the *document number*, *document type*, *document part*, and *document version.* The SAP interface program then sends the existing document data to the external application program as defined in settings in customizing. At the same time, a lock entry is created in the SAP System to prevent concurrent access by other users.

The changes made to the document data in the external system are returned to the SAP System in the next communication step. Finally, the SAP interface program ensures that the changes are saved in the SAP System.

Changing a Document Info Record With Classification

This function is identified by sub-process code *MC*.

The function allows you to change document info record data as in function *DOCCH MR.* You can also maintain classification data. You have the following options:

- changing an existing allocation and/or existing characteristic values
- creating an allocation to a new class
- deleting the allocation to a class

You can only create, change, or delete the allocation to one class in one function call. You cannot process several class allocations in one function.

Please note that classification is connected to engineering change management. For this reason, all changes to classification data can be made historically with either a date or a change number.

- The communication flow for function *DOCCH MC* is the same as for function *DOCCH MR* up to communication step *DOCCH MD* (SAP→CAD).
- When you send the changed document info record data to the SAP System in the next communication step *DOCCH MD* (CAD→SAP), you can also send values for the *class type* and *class* whose allocation you want to change.

  This data has a restrictive function.

  - If you do not enter these values, a data string including all classes to which the document is allocated is sent to the external application program.
  - If you enter a value for the class type, only the classes of this class type are sent to the external application program.
  - There is also a further restriction you can make regarding the standard class. However, if a standard class is defined for the document, or if the document is only allocated to one class, the communication step for transferring a list of classes is no longer necessary. In this case, the information on characteristics and assigned values is sent directly to the external program.

The communication flow of function *DOCCH MC* when changing an existing allocation to a class is described in the table below.

Communication flow for function DOCCH MC when changing an allocation to a class

SP code	Direction	Data transferred	Length
MC	→	Document key data as defined in table TCIU	variable
MD	←	Document data in the SAP System as defined in table TCID	variable
MD	→	Changed document data as defined in table TCIU including class type, class, and scope	variable

Changing a Document Info Record With Classification

RC	←	List of All Classes [Seite 126], in which the document is classified	variable
RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

After communication step *DOCCH CR* (SAP→CAD), you can either assign values to the characteristics or continue with a changed combination of class type, class, and scope.

Communication step *DOCCH RC* (SAP→CAD) occurs if the restrictions defined in communication step *DOCCH MD* (CAD→SAP) find more than one class in which the document is classified.

If the change to the classification data is to create an allocation to an additional class, the communication flow of the table above is changed as shown below:

Communication flow for function DOCCH MC when creating an allocation to a new class

SP code	Direction	Data transferred	Length
MC	→	Document key data as defined in table TCIU	variable
MD	←	Document data in the SAP System as defined in table TCID	variable
MD	→	Changed document data as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], in which the document is classified	variable

NC	→	New class type &) New class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

If you want to delete an existing allocation to a class, the communication flow is as follows:

Communication flow for function DOCCH MC when deleting an allocation to a class

SP code	Direction	Data transferred	Length
MC	→	Document key data as defined in table TCIU	variable
MD	←	Document data in the SAP System as defined in table TCID	variable
MD	→	Changed document data as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], in which the document is classified	variable
DC	→	Class type to be deleted &) Class to be deleted &)	variable variable
MC	←	Function message	variable

Changing a Document Info Record (Object Links Only)

This function is identified by sub-process code *OB*.

The function allows you to change which SAP objects are linked to a document info record. You cannot make changes to the document info record itself using this function.

- First, you send the key of the document to be changed to the SAP System in function call *DOCCH OB*.
- The SAP interface program then sends the list of objects that can be linked to documents of the relevant document type in communication step *DOCCH DO*. Each object type has a consecutive number, which is used to process further communication. Each object type for which objects are linked to the relevant document has an indicator.

  In communication step *DOCCH DO,* the consecutive number of the SAP object for which you want to create, change, or delete a link to the document is sent to the SAP System.
- The SAP interface program then sends the key fields of the selected object type and the list of objects of this object type that are already linked to the document info record in the two sub-functions *DOCCH ST* and *DOCCH OL*.
  - In sub-function *DOCCH ST,* the length of the individual object key fields is also sent. You must keep to these lengths when you link objects to the document. In addition, the description of each object already linked to the document is displayed.
  - In communication step *DOCCH OL,* the changed object links of the current object type are sent to the SAP System.
    - You must also send all objects that are already linked to the document to the SAP System, if you want the link to remain.
    - The link is deleted for any objects that are not sent to the SAP System.

    The SAP System makes the change to object links and sends a message to the external application program.

    You can only change object links of one object type in one function call. If you want to change object links of more than one object type, you have to call the function several times.

The communication flow of function *DOCCH OB* is described in the table below.

Communication flow for function DOCCH OB

SP code	Direction	Data transferred	Length
OB	→	Document number &)	variable
		Document type &)	variable
		Document part &)	variable
		Document version &)	variable

DO	←	Number of first object type &)	variable
		Description of last object type &)	variable
		Indicator: object links exist &)	variable
		.	.
		.	.
		.	.
		Number of last object type &)	variable
		Description of last object type &)	variable
		Indicator: object links exist &)	variable
DO	→	Number of object type required	variable
ST	←	First key field of object type &)	variable
		Length of first key field &)	variable
		.	.
		.	.
		.	.
		Last key field of object type &)	variable
		Length of last key field &)	variable
		Heading for text field &)	variable
		Length of text field &)	variable
OL	←	First key field of first object &)	variable
		.	.
		.	.
		.	.
		Last key field of first object &)	variable
		Description of first object &)	variable
		.	.
		.	.
		.	.
		First key field of last object &)	variable
		.	.
		.	.
		.	.
		Last key field of last object &)	variable
		Description of last object &)	variable

Changing a Document Info Record (Object Links Only)

OL	→	First key field of first object . . . Last key field of first object . . . First key field of last object . . . Last key field of last object	variable . . . variable . . . variable . . . variable
OB	←	Function message	variable

This function is called from the external application program as follows:

hd = SapDocch(scode, sndstr, rcvstr, msg).

DOCRQ - Displaying a Document Info Record

The following functions for displaying document info records are supported:

- Displaying a Document Info Record Without Classification [Seite 152]
- Displaying a Document Info Record With Classification [Seite 153]
- Finding a Document Info Record via Matchcode [Seite 155]
- Finding a Document Info Record Using Transaction CV04 [Seite 158]
- Displaying a Document Info Record (Object Links Only) [Seite 161]
- Displaying the Status History of a Document Info Record [Seite 164]

Displaying a Document Info Record Without Classification

This function is identified by sub-process code *MR*.

It allows you to display data in a document info record, with the exception of classification data and data on object links. The communication flow of function *DOCRQ MR* is described in the table below.

Communication flow for function DOCRQ MR

SP code	Direction	Data transferred	Length
MR	→	Document key data as defined in table TCIU	variable
MR	←	Document data in the SAP System as defined in table TCID	variable

When you call the function, the data that identifies the document info record is sent to the SAP System. This data comprises the document number, document type, document part, and document version. The SAP interface program then sends the existing document data to the external application program as defined in settings in customizing.

Displaying a Document Info Record With Classification

This function is identified by sub-process code *MC*.

The function allows you to display data in the document info record as in function *DOCRQ MR*. You can also display classification data. The communication flow of function *DOCRQ MC* is described in the table below.

Communication flow for function DOCRQ MC

SP code	Direction	Data transferred	Length
MC	→	Document key data as defined in table TCIU including class type, class, and scope	variable
MC	←	Document data in the SAP System as defined in table TCID	variable
RC	←	List of All Classes [Seite 126], in which the document is classified	variable
RC	→	Class type &)	variable
		Class &)	variable
		Indicator: required characteristics &)	variable
		Indicator: optional characteristics &)	variable
		Indicator: characteristics with assigned values &)	variable
		Indicator: characteristics without assigned values &)	variable
		Indicator: inherited characteristics &)	variable
		Indicator: standard class &)	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
RC	→	Class type &)	variable
		Class &)	variable
		Indicator: required characteristics &)	variable
		Indicator: optional characteristics &)	variable
		Indicator: characteristics with assigned values &)	variable
		Indicator: characteristics without assigned values &)	variable
		Indicator: inherited characteristics &)	variable
		Indicator: standard class &)	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Assigned Characteristic Values [Seite 120]	variable

Displaying a Document Info Record With Classification

When you call function *DOCRQ MC,* you can send values for the *class type* and *class* whose allocation you want to display.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes to which the document is allocated is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class. However, if a standard class is defined for the document, or if the document is only allocated to one class, the communication step for transferring a list of classes is no longer necessary. In this case, the information on characteristics and assigned values is sent directly to the external program.
- You can also enter a date or a change number. Since classification is made with history, the status on a specific date is displayed. If you do not enter a date, the current date is taken as a default.
  - If the restrictions on classification that you send with the function call are such that **several** classes are selected, this list of classes is first sent to the external application program in communication step *DOCRQ RC* (SAP→CAD). In the external system, you select the class you are interested in and return the selected class to the SAP System in communication step *DOCRQ RC* (CAD→SAP).

    You then see the characteristic and characteristic values for the class and scope you require.

    You can repeat communication step *DOCRQ RC* (CAD→SAP) as many times as you like. This allows you to display the classification data of the document in each class in the list and to vary the scope. Since there is no obvious point at which to end function *DOCRQ MC,* please note that you must use the *BREAK* function to terminate this function.
  - If only **one** class is selected or if the document is classified in exactly one class, you can only vary the scope in communication step *DOCRQ RC* (CAD→SAP).

In function *DOCRQ RC* (CAD→SAP), you can only display the classification data that belongs to classes selected when you called the function. Classification data for classes that were not selected cannot be displayed.

This means that, if the document is classified in classes of two different class types, and you enter a class type when you call function *DOCRQ MC* (CAD→SAP), you do not see the allocations to classes of the second class type. You cannot display these allocations in function *DOCRQ RC* (CAD→SAP) and by entering the other class type, either. You need to call function *DOCRQ MC* again.

Finding a Document Info Record via Matchcode

This function is identified by sub-process code *RL*.

The function allows you to find a document in the SAP System if you do not know the key fields that identify the document but do know other master data (such as the description or the user who is responsible for the document). You also have the option of using the special document retrieval functions *DOCRQ DL* and *DOCRQ OS,* or the find objects function in the classification system. For more information on finding objects via classification, see.

The matchcode functionality allows you to define a matchcode ID such that it starts the function for finding objects via classes or a report, instead of the standard matchcode search function. These matchcode IDs cannot be processed via the CAD interface. If you select one of these matchcode IDs, you see an error message telling you that the matchcode ID cannot be used.

When you use a matchcode, either:

- You do not know the field structure of the selected matchcode ID, or
- You know the field structure of the selected matchcode ID.

The communication flow of function *DOCRQ RL* is described in the tables below.

Communication flow for function DOCRQ RL if you do not know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Selected matchcode ID &) Matchcode sub-ID &)	variable variable
RF	←	 First matchcode field &) Length of first matchcode field &) . . . Last matchcode field &) Length of last matchcode field &)	variable variable variable . . . variable variable
RF	→	Matchcode selection string (Matchcode fields are separated by a point)	variable
CL	←	Document data in the SAP System as defined in table TCID	variable
CL	→	None	0

Finding a Document Info Record via Matchcode

RL	←	Document data in the SAP System as defined in table TCID	variable

Communication flow for function DOCRQ RL if you know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Matchcode selection string	variable
CL	←	Document data in the SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Document data in the SAP System as defined in table TCID	variable

The matchcode sub-ID is still included in the transferred data strings to ensure compatibility with R/2. In data strings sent from the R/3 System to the external application program, the matchcode sub-ID is SPACE. In data strings sent from the external application program to the SAP System, you can either send the additional separator character or not.

- After you call function *DOCRQ RL,* first all the matchcode IDs defined in the SAP document management system are sent to the external application program.
- For further processing, there is a difference depending on whether you know the field structure of the selected matchcode or not.
  - If you do **not** know the field structure, you send the selected matchcode ID to the SAP System in the next communication step in order to get the field structure.
  - However, if you know the field structure, you can use the next communication step to send the matchcode selection string. To do this, you use the matchcode identifier (usually the '0' character). The matchcode identifier is transferred to the external application program using system function *SYSRQ PA*.

When you find documents via matchcode, please note that the master record is read for each document info record found for your selection criteria in order to find the data that is not included in the matchcode records. However, this can lead to runtime problems if the selection criteria are not specific enough, so that a large number of document info records is found.

For this reason, control table TCIS contains parameter TCIS-MCMAX, which defines the maximum number of matchcode records, after which communication with the external application program is triggered. If such a communication is necessary, you use sub-process code *CL*.

If you call function *DOCRQ CL,* you can then request the following matchcode records.

As of release 3.0, all matchcode selection functions were changed such that functions *DOCRQ ID* and *DOCRQ RF* can be called in any communication step. This has the advantage that you do not need to start again by calling function *DOCRQ RL* if your matchcode search is unsuccessful. Instead, you can enter new values in the matchcode fields directly if you want to use the current matchcode ID, or switch to another matchcode ID. However, this means that the matchcode selection functions no longer come to a natural end. As of Release 3.0, you terminate these functions using the *BREAK* function

Finding a Document Info Record Using Transaction CV04

There are two variants of this function, which are identified by sub-process codes *DL* and *OS*.

These functions allow you to find document info records via retrieval function CV04 *(Find document)*, which is defined in the SAP System specifically for this purpose. The difference between the two variants *DOCRQ DL* and *DOCRQ OS* is that *DOCRQ OS* also allows you to find documents via the objects linked to them. Apart from this, the two functions are identical. The communication flow of function *DOCRQ DL* and *DOCRQ OS* is described in the tables below.

Communication flow for function DOCRQ DL

SP code	Direction	Data transferred	Length
DL	→	Document data as defined in table TCIU	variable
TB	→	Language indicator &)	variable
		First catchword &)	variable
		Operator (AND, NAND, OR) &)	variable
		.	.
		.	.
		.	.
		Last catchword &)	variable
DL	←	Document data in the SAP System as defined in table TCID	variable

When you call function *DOCRQ DL* or *DOCRQ OS,* the search criteria as defined in Customizing are sent to the SAP System.

In the following record, you can send search strings for long texts. This is optional.

- If you do not want to search for strings in long texts, use data length ‘0’ in function call *DOCRQ TB*.
- If you want to search for strings in long texts, you can use up to seven strings, linked with ‘AND’ or ‘OR’ with a language indicator. You also have the option of using three more strings linked with ‘NAND’. The ‘NAND’ operator excludes documents with a long text containing these strings.

If you are using function *DOCRQ DL,* you now start the find document function. However, if you are using function *DOCRQ OS,* you can send the object types that can be linked to the document type to the external application program. This allows you to use the SAP objects linked to documents as additional search criteria. However, you can only use objects of one object type. (In the online SAP System, you can use more than one object type.) This means that you can search for all documents of a document type that are linked to materials 4711 and 4712, but you cannot search for all documents of a document type that are linked to material 4711 and equipment 4712.

Another restriction is that you cannot enter a generic document type in these two functions. You must always enter the document type in full.

- In the list of SAP object types sent to the external application program, each object type has a consecutive number. You send the number of the object type you require to the SAP System in communication step *DOCRQ OR*.
- The SAP System then sends the key fields of the object type (including their lengths) to the external application program in function *DOCRQ KF*.
- Now the external application program can send the list of linked objects to the SAP System. Please note that the individual fields must be entered in the predefined length and without separator characters.

Once the find document function has been executed, the fields of the document info records found are sent to the external application program as defined in table TCID.

Communication flow for function DOCRQ OS

SP code	Direction	Data transferred	Length
OS	→	Document data as defined in table TCIU	variable
TB	→	Language indicator &)	variable
		First catchword &)	variable
		Operator (AND, NAND, OR) &)	variable
		.	.
		.	.
		.	.
		Last catchword &)	variable
OR	←	Number of first object type &)	variable
		Description of last object type &)	variable
		.	.
		.	.
		.	.
		Number of last object type &)	variable
		Description of last object type &)	variable
OR	→	Number of object type required	variable
KF	←	First key field of object type &)	variable
		Length of first key field &)	variable
		.	.
		.	.
		.	.
		Last key field of object type &)	variable
		Length of last key field &)	variable

Finding a Document Info Record Using Transaction CV04

KF	→	First key field of first object . . . Last key field of first object . . . First key field of last object . . . Last key field of last object	variable . . . variable . . . variable . . . variable
OS	←	Document data in the SAP System as defined in table TCID	variable

Displaying a Document Info Record (Object Links Only)

This function is identified by sub-process code *OB*.

The function allows you to display the SAP objects linked to a document info record. You cannot display other data in the document info record with this function.

- When you call function *DOCRQ OB,* first you send the key of the document you want to display to the SAP System.
- The SAP interface program then sends a list of object types that can be linked to documents of the relevant document type in communication step *DOCRQ DO*. Each object type has a consecutive number, which is used to process further communication. Each object type for which objects are linked to the relevant document has an indicator.
- In communication step *DOCRQ DO,* the consecutive number of the SAP object type for which you want to display object links is sent to the SAP System.

  The SAP interface program then sends the key fields of the selected object type, and a list of objects of this object type that are linked to the relevant document, using the two sub-functions *DOCRQ ST* and *DOCRQ OL*.

  - In sub-function *DOCRQ ST,* the lengths of the individual object key fields are also sent. In addition, the description of each object is displayed in the list of linked objects.
  - You can repeat function call *DOCRQ DO* as often as you like, in order to display the linked objects of other object types. This mans that function *DOCRQ OB* does not come to a natural end. You have to terminate the function with function call *BREAK* when you want to exit.

The communication flow of function *DOCRQ OB* is described in the table below.

Communication flow for function DOCRQ OB

SP code	Direction	Data transferred	Length
OB	→	Document number &)	variable
		Document type &)	variable
		Document part &)	variable
		Document version &)	variable

Displaying a Document Info Record (Object Links Only)

DO	←	Number of first object type &)	variable
		Description of last object type &)	variable
		Indicator: object links exist &)	variable
		.	.
		.	.
		.	.
		Number of last object type &)	variable
		Description of last object type &)	variable
		Indicator: object links exist &)	variable
DO	→	Number of object type required	variable
ST	←	First key field of object type &)	variable
		Length of first key field &)	variable
		.	.
		.	.
		.	.
		Last key field of object type &)	variable
		Length of last key field &)	variable
		Heading for text field &)	variable
		Length of text field &)	variable
OL	←	First key field of first object &)	variable
		.	.
		.	.
		.	.
		Last key field of first object &)	variable
		Description of first object &)	variable
		.	.
		.	.
		.	.
		First key field of last object &)	variable
		.	.
		.	.
		.	.
		Last key field of last object &)	variable
		Description of last object &)	variable

DO	→	Number of object type required	variable
ST	←	First key field of object type &)	variable
		Length of first key field &)	variable
		.	.
		.	.
		.	.
		Last key field of object type &)	variable
		Length of last key field &)	variable
		Heading for text field &)	variable
		Length of text field &)	variable
OL	←	First key field of first object &)	variable
		.	.
		.	.
		.	.
		Last key field of first object &)	variable
		Description of first object &)	variable
		.	.
		.	.
		.	.
		First key field of last object &)	variable
		.	.
		.	.
		.	.
		Last key field of last object &)	variable
		Description of last object &)	variable

Displaying the Status History of a Document Info Record

This function is identified by sub-process code *HS*.

The function allows you to display the status history of a document info record. You cannot display other data in the document info record with this function.

When you call function *DOCRQ HS,* first the key of the document you want to display is sent to the SAP System. The SAP interface program then sends a list of statuses that the document has had so far in its life cycle in communication step *DOCRQ HS*.

The communication flow of function *DOCRQ HS* is described in the table below.

Communication flow for function DOCRQ HS

SP code	Direction	Data transferred	Length
HS	→	Document number &)	variable
		Document type &)	variable
		Document part &)	variable
		Document version &)	variable
HS	←	Language-dependent first document status &)	variable
		Status text &)	variable
		Status type &)	variable
		Indicator: released status &)	variable
		Indicator: completion message for change &)	variable
		Date on which status was set &)	variable
		Time at which status was set &)	variable
		User who set the status &)	variable
		Log line &)	variable
		.	.
		.	.
		.	.
		Language-dependent last document status &)	variable
		Status text &)	variable
		Status type &)	variable
		Indicator: released status &)	variable
		Indicator: completion message for change &)	variable
		Date on which status was set &)	variable
		Time at which status was set &)	variable
		User who set the status &)	variable
		Log line &)	variable

This function is called from the external application program as follows:

hd = SapDocrq(scode, sndstr, rcvstr, msg).

DOCCI - *Check In* Functions for Original Application Files

Note: This function can only be used for document types that do **not** us the Knowledge Provider for storing original application files. This setting is done in Customizing of *Document Management*, work step Define document type.

As of release 3.0, you can subject original application files for a document info record to the access control of the SAP document management system using the check-in function in the CAD interface. The storage location is only known to the document management system, so the original application files can then only be displayed or edited via the SAP document management system. You can no longer access the files directly.

The following storage locations are supported for original application files:

- **Vault**

  Server mounted onto the application server that the SAP System runs on. This server is protected from direct access by the file system.

  As of release 3.0E, you also have the option of defining any computer in your network as a vault. This server need not be mounted on the application sever. Files can be transported directly between a frontend computer and the vault. This speeds up transfer.

- **SAP System**
- **Archive**

The following check-in functions are supported for original application files:

- Check In for Original Application Files [Seite 166]
- Displaying a List of All Available Vaults [Seite 168]

Check In for Original Application Files

This function is identified by sub-process code *MR*.

The function allows you to *check in* the original application file associated with the document info record. The communication flow of the *DOCCI MR* function is described in the table below.

Communication flow for the DOCCI MR function

SP code	Direction	Data transferred	Length
MR	→	Document number &)	variable
		Document type &)	variable
		Document part &)	variable
		Document version &)	variable
		Application number (‘1’ or ‘2’) &)	variable
		Document status after check in &)	variable
		Log entry &)	variable
		Data carrier of vault &)	variable
		Access path of file to be checked in &)	variable
MR	←	Function message	variable

When you call the function, send the document key and the number of the application. The possible entries ‘**1**’ and ‘**2**’ refer to the two original application files that can be linked to the document info record.

You also send the document status to be set, together with any log entry you require.

Finally, you identify the storage location where the “original” document is to be saved after c*heck in*. You must enter the original application file and a complete access path.

This access path cannot be longer than 70 characters.

You have the following options:

- **SAP System**

  The original file is saved to the SAP database.

- **Archive**

  The original file is saved to the archive.

- **Vault name**

  The original file is saved to the vault you enter.

  You can request a list of available vaults using the *DOCCI CT* function.

As of release 3.0E, you have the option of transferring files directly between the local computer and the vault. This eliminates the load on the application server. You can activate this function in Customizing for Document Management. Before you can use the function, program SAPFTP (UNIX) or SAPFTP.EXE (WINDOWS/NT) must be available on the local computer.

The function looks for program SAPFTP or SAPFTP.EXE in the following directories in the following sequence:

- Path defined with environment variable CADRFC
- Start path for program

  (This is determined from parameter ARGV, which is transferred with function call SapConnc or SapConncRfc.)
- Working directory

Displaying a List of All Available Vaults

This function is identified by sub-process code *CT*.

The function gives you a list of all the vaults defined in the SAP document management system. The communication flow of function *DOCCI CT* is described in the table below.

Communication flow for function DOCCI CT

SP code	Direction	Data transferred	Length
CT	→	None	0
CT	←	Name of first vault &) Description of vault &) . . . Name of last vault &) Description of vault &)	variable variable . . . variable variable

This function is called from the external application program as follows:

hd = SapDocci(scode, sndstr, rcvstr, msg).

DOCCO - Check Out Functions for Original Application Files

If an original application file was checked in with function *DOCCI MR*, the file is subject to SAP document management system control. The original application file can then no longer be edited without going via the document management system (check out), because the storage location of the original application file is only availabe in the document management system.

If you want to edit the original application file, you have to check it out using function *DOCCO MR*. This function requests a copy of the original application file. This copy is saved to a local directory. At the same time, the status of the document info record is set such that no other user can check this document out. The original application file can then be edited locally and checked in after the change.

The following check out functions are supported for original application files:

- Check Out for Original Application Files [Seite 170]
- Displaying an Original Application File [Seite 172]

Check Out for Original Application Files

This function is identified by sub-process code *MR*.

The function allows you to *check out* the original application file associated with the relevant document info record. The communication flow of the *DOCCO MR* function is described in the table below.

Communication flow for the DOCCO MR function

SP code	Direction	Data transferred	Length
MR	→	Document number &)	variable
		Document type &)	variable
		Document part &)	variable
		Document version &)	variable
		Application number (‘1’ or ‘2’) &)	variable
		Log entry &)	variable
RF	←	Data carrier &)	variable
		Original data carrier &)	variable
		Original path name &)	variable
RF	→	New data carrier &)	variable
		New path name &)	variable
MR	←	Function message	variable
		Access path for original document	

When you call the function, send the document key and the number of the application. The possible entries ‘**1**’ and ‘**2**’ refer to the two original application files that can be linked to the document info record. You also send the log entry you want to set, if any, when the document status is changed.

The function flow from here depends on whether the original application file is available in the file system or was checked in previously.

- If the original application file is available in the file system, the status of the document info record is changed.

  In addition, the absolute access path to the original application file is sent to the external application program in function call *DOCCO* MR. The intermediate step *DOCCO RF,* shown in the table above, is not required and is therefore skipped.

- If the original application file was checked in previously, this is communicated to the external application system in communication step *DOCCO RF*. In this case, an additional communication step is required, because the external application program has to communicate which (local) directory the checked out copy of the original application file is to be saved to. In communication step *DOCCO RF,* the current data carrier, the original data carrier, and the original path name are sent to the external application program.

The current data carrier can assume one of the following values:

- **SAP System**

  The original application file is stored in the SAP database.

- **Archive**

  The original application file is stored in the archive.

- **Vault name**

  The original application file is stored in the vault you enter.

The values sent for the original data carrier and the original path name are to be seen as default values. They can be overwritten with another combination of data carrier and path name. The values for data carrier and path name sent to the SAP System in communication step *DOCCO RF* determine the storage location of the checked out copy of the original application file.

This access path cannot be longer than 70 characters.

The SAP-CAD interface program then triggers the *check out* of the original application file and makes the relevant change to the document info record. If the *check out* cannot be executed using the path name entered, for example, due to missing write authorization or because the current document status does not allow *check out,* you see an error message.

As of release 3.0E, you have the option of transferring files directly between the local computer and the vault. This eliminates the load on the application server. You can activate this function in Customizing for Document Management. Before you can use the function, program SAPFTP (UNIX) or SAPFTP.EXE (WINDOWS/NT) must be available on the local computer.

The function looks for program SAPFTP or SAPFTP.EXE in the following directories in the following sequence:

- Path defined with environment variable CADRFC
- Start path for program

  (This is determined from parameter ARGV, which is transferred with function call SapConnc or SapConncRfc.)

- Working directory

Displaying an Original Application File

This function is identified by sub-process code *OR*.

The function allows you to access the original application file that is associated with the relevant document info record. The communication flow of function *DOCCO OR* is described in the table below.

Communication flow for function DOCCO OR

SP code	Direction	Data transferred	Length
OR	→	Document number &)	variable
		Document type &)	variable
		Document part &)	variable
		Document version &)	variable
		Application number (‘1’ or ‘2’) &)	variable
OR	←	Access path for original document	variable
MR	←	Access path for original document	variable

When you call the function, send the document key and the number of the application. The possible entries ‘**1**’ and ‘**2**’ refer to the two original application files that can be linked to the document info record.

You must also decide whether the original application file is available in the file system or has already been checked in. If the original application file is available in the file system, the status of the document info record is changed. In addition, the absolute access path to the original application file is sent to the external application program in function call *DOCCO*.

When the document original application file was already checked in the SAP System automatically assigns a name. This file name is used for a copy of the original application file.

The current data carrier can assume one of the following values:

- **SAP System**

  The original application file is stored in the SAP database.

- **Archive**

  The original application file is stored in the archive.

- **Vault name**

  The original application file is stored in the vault you enter.

The values sent for the original data carrier and the original path name are to be seen as default values. They can be overwritten with another combination of data carrier and path name. The values for data carrier and path name sent to the SAP System in communication step *DOCCO RF* determine the storage location of the checked out copy of the original application file.

The SAP-CAD interface program then makes the binary data of the original document available under the path name entered. If the data cannot be stored under the path name entered due to missing write authorization; for example, you see an error message.

As of release 3.0E, you have the option of transferring files directly between the local computer and the vault. This eliminates the load on the application server. You can activate this function in Customizing for the Document Management System. Before you can use the function, program SAPFTP (UNIX) or SAPFTP.EXE (WINDOWS/NT) must be available on the local computer.

The function looks for program SAPFTP or SAPFTP.EXE in the following directories in the following sequence:

- Path defined with environment variable CADRFC
- Start path for program

  (This is determined from parameter ARGV, which is transferred with function call SapConnc or SapConncRfc.)

  - Working directory

Functions for BOMs

The CAD interface supports the following functions for processing BOMs (bills of material):

BOMCR – Creating a BOM [Seite 175]

BOMCN – Changing a BOM [Seite 177]

BOMRQ – Displaying a BOM [Seite 179]

BOMWU – Displaying a Where-Used List [Seite 181]

CadBomCreateSubItem – Creating a BOM With Sub-Items [Seite 182]

CadBomChangeSubItemRead – Reading and Changing a BOM [Seite 183]

CadBomChangeSubItem – Changing a BOM with Sub-Items [Seite 184]

CadBomDisplaySubItem – Displaying a BOM With Sub-Items [Seite 185]

BOMCR – Creating a BOM

The CAD interface supports 2 functions for transferring BOMs from an external system to the SAP System:

- Function *BOMCR MP*

  In this function, item numbers for the individual BOM items must be sent from the external application program.

- Function *BOMCR MR*

  In this function, you **cannot** use item numbers. The item numbers are assigned automatically by the SAP System.

To transfer a BOM to the SAP System, two **sub-functions** are required:

- The first sub-function transfers the BOM header data.
- The second sub-function transfers the data for all BOM items.

All the data to be transferred must be defined in Customizing for the CAD interface.

The communication flow for functions *BOMCR MP* and *BOMCR MR* is described in the tables below.

Communication flow for function BOMCR MP

SP code	Direction	Data transferred	Length
MP	→	BOM header data as defined in table TCIU	variable
SP	→	BOM item data as defined in table TCIU with item numbers	variable
MP	←	Function message	variable

Communication flow for function BOMCR MR

SP code	Direction	Data transferred	Length
MR	→	BOM header data as defined in table TCIU	variable
SR	→	BOM item data as defined in table TCIU without item numbers	variable
MR	←	Function message	variable

When you use function *BOMCR MP,* the individual BOM items are automatically sorted in ascending order of item numbers. However, when you use function *BOMCR MR,* there are no item numbers to sort on, so you have various options for sorting the BOM items. Control table TCIS contains parameter TCIS-SORTK, which you can use to control the sort sequence for BOM items.

Parameter TCIS-SORTK can have the following values:

BOMCR – Creating a BOM

- **N**

  The items are not sorted. The BOM items are left in the sequence in which they are transferred.

- **K**

  The items are sorted according to component number, item category, and sort string, respectively.

- **S**

  The items are sorted according to sort string, component number, and item category, respectively.

When you use function *BOMCR MR,* parameter TCSPR-PSINC in table TCSPR is used to determine the item number increment. If you do not define this parameter, the default value ‘0010’ is used.

This function is called from the external application program as follows:

hd = SapBomcr(scode, sndstr, rcvstr, msg).

BOMCN – Changing a BOM

The CAD interface supports two functions for changing BOMs:

- Function *BOMCN MP*

  In this function, item numbers for the individual BOM items must be sent from the external application program.

- Function *BOMCN MR*

  In this function, you **cannot** use item numbers.

To make a change to a BOM via the CAD interface, you must send the material number, the BOM usage, and the date of the change, when you call the function. You can also send a change number or a revision level instead of a date. In this case, the SAP System determines the date. The status of the BOM on the relevant date is determined, and the BOM data is sent to the external application program in 2 data strings. The changed BOM is then transferred back to the SAP System. The SAP interface program compares the old BOM with the changed BOM. Finally, the change is made using the BOM transaction.

When you change a BOM via the CAD interface, you can only change items that were created via the CAD interface. All of these BOM items are marked with a CAD indicator. When you change a BOM via the CAD interface, only these items are transferred to the external application program. Any other items created in the SAP System at a later date are ignored.

The communication flow of functions *BOMCN MP* and *BOMCN SR* is described in the tables below.

Communication flow for function BOMCN MP

SP code	Direction	Data transferred	Length
MP	→	Data identifying the BOM as defined in table TCIU	variable
MK	←	Existing BOM header data as defined in table TCID	variable
SP	←	Existing BOM item data as defined in table TCID	variable
MK	→	Changed BOM header data as defined in table TCIU	variable
SP	→	Changed BOM item data as defined in table TCIU with item numbers	variable
MP	←	Function message	variable

BOMCN – Changing a BOM

Communication flow for function BOMCN MR

SP code	Direction	Data transferred	Length
MR	→	Data identifying the BOM as defined in table TCIU	variable
MK	←	Existing BOM header data as defined in table TCID	variable
SR	←	Existing BOM item data as defined in table TCID	variable
MK	→	Changed BOM header data as defined in table TCIU	variable
SR	→	Changed BOM item data as defined in table TCIU without item numbers	variable
MR	←	Function message	variable

When you use *BOMCN MP,* the individual BOM items are automatically sorted in ascending order of item numbers. However, when you use function *BOMCN MR,* there are no item numbers to sort on, so you have various options for sorting the BOM items. Control table TCIS contains parameter TCIS-SORTK, which you can use to control the sort sequence for BOM items.

Parameter TCIS-SORTK can have the following values:

- **N**

  The items are not sorted. The BOM items are left in the sequence in which they are transferred.

- **K**

  The items are sorted according to component number, item category, and sort string, respectively.

- **S**

  The items are sorted according to sort string, component number, and item category, respectively.

When you use function *BOMCN MR,* parameter TCSPR-PSINC in table TCSPR is used to determine the item number increment. If you do not define this parameter, the default value ‘0010’ is used.

This function is called from the external application program as follows:

hd = SapBomcn(scode, sndstr, rcvstr, msg).

BOMRQ – Displaying a BOM

The CAD interface supports the following functions for displaying a BOM:

- *BOMRQ MR* – Display BOM (single-level)
- *BOMRQ LB* – Display BOM level by level (multi-level)
- *BOMRQ MB* – Display multi-level BOM
- *BOMRQ QB* – Display summarized BOM

When you call the individual functions, you can define restrictions for the BOM items you want to display. For example, you can define that you only want to display BOM items of a specific item category.

The communication flow of functions *BOMRQ MR, BOMRQ LB, BOMRQ MB,* and *BOMRQ QB* is described in the tables below.

Communication flow for function BOMRQ MR

SP code	Direction	Data transferred	Length
MR	→	BOM selection data as defined in table TCIU	variable
MR	←	BOM header data as defined in table TCID	variable
SP	←	BOM item data as defined in table TCID	variable

Communication flow for function BOMRQ LB

SP code	Direction	Data transferred	Length
LB	→	BOM selection data as defined in table TCIU	variable
LB	←	BOM header data as defined in table TCID	variable
LP	←	BOM item data as defined in table TCID	variable

Communication flow for function BOMRQ MB

SP code	Direction	Data transferred	Length

BOMRQ – Displaying a BOM

MB	→	BOM selection data as defined in table TCIU	variable
MB	←	BOM header data as defined in table TCID	variable
MP	←	BOM item data as defined in table TCID	variable

Communication flow for function BOMRQ QB

SP code	Direction	Data transferred	Length
QB	→	BOM selection data as defined in table TCIU	variable
QB	←	BOM header data as defined in table TCID	variable
QP	←	BOM item data as defined in table TCID	variable

This function is called from the external application program as follows:

hd = SapBomrq(scode, sndstr, rcvstr, msg).

BOMWU – Displaying a Where-Used List

This function allows you to produce a where-used list via the CAD interface. If you want to change a part in the external system, you can display a list of materials whose BOMs contain this part.

The communication flow of function *BOMWU RL* is described in the table below:

Communication flow for function BOMWU RL

SP code	Direction	Data transferred	Length
RL	→	BOM selection data as defined in table TCIU	variable
RL	←	BOM header data as defined in table TCID	variable

This function is called from the external application program as follows:

hd = SapBomwu(scode, sndstr, rcvstr, msg).

CadBomCreateSubItem – Creating a BOM With Sub-Items

You can create a BOM with sub-items in the background with the function CadBomCreateSubItem. You use this function in the same way as dialog RFC functions. The only difference is that no SAP dialog screen is processed.

You can write item and sub-item data to internal tables with the functions CadRfcWriteBomTabEntry and CadRfcWriteBomSubItemEntry. These tables are transferred to the SAP System when you call the function CadBomCreateSubItem.

You can find an example of this function call in file CADRFTST.C.

C function call:	*int CadBomCreateSubItem (BICSK *MaterialHeaderData, CadRfcImportField *ImportData, ITAB_H*FieldData, ITAB_H *DmsClassData);*
SAP function module:	CAD_CREATE_BOM_WITH_SUB_ITEMS

See also:

Filling out the BOM Item Table [Seite 582]

Filling out the BOM Sub-Item Table [Extern]

CadBomChangeSubItemRead – Reading and Changing a BOM

This function returns the current status of a BOM with sub-items to the external system and locks the BOM so that it can be changed. The changed BOM data must be transferred to the SAP System with the function CadBomChangeSubItem.

You can read item and sub-item data with the functions CadRfcGetBomItemEntry and CadRfcGetBomSubItemEntry.

You can find an example of this function call in file CADRFTST.C.

C function call:	*int CadBomChangeSubItemRead(BICSK *MaterialHeaderData, CadRfcImportField *ImportData, ITAB_H*FieldData, ITAB_H *DmsClassData);*
SAP function module:	CAD_DISPLAY_BOM_WITH_SUB_ITEMS

See also:

Reading the BOM Item Table [Seite 583]

Reading the BOM Sub-Item Table [Extern]

CadBomChangeSubItem – Changing a BOM with Sub-Items

You can change BOMs with sub-items with the function CadBomChangeSubItem. Before you call this function, the BOM must be read with the function CadBomChangeSubItemRead. At the same time, the BOM is locked against any other changes.

You can write item and sub-item data to internal tables with the functions CadRfcWriteBomTabEntry and CadRfcWriteBomSubItemEntry. These tables are transferred to the SAP System when you call the function CadBomChangeSubItem.

You can find an example of this function call in file CADRFTST.C.

C function call:	*int CadBomChangeSubItem (BICSK *MaterialHeaderData, CadRfcImportField *ImportData, ITAB_H*FieldData, ITAB_H *DmsClassData);*
SAP function module:	CAD_CHANGE_BOM_WITH_SUB_ITEMS

See also:

Filling out the BOM Item Table [Seite 582]

Filling out the BOM Sub-Item Table [Extern]

CadBomDisplaySubItem – Displaying a BOM With Sub-Items

You can read BOMs with sub-items with the function CadBomDisplaySubItem. You can access the item and sub-item data with the functions CadRfcGetBomItemEntry and CadRfcGetBomSubItemEntry.

You can find an example of this function call in file CADRFTST.C.

C function call:	*int CadBomDisplaySubItem(BICSK *MaterialHeaderData, CadRfcImportField *ImportData, ITAB_H*FieldData, ITAB_H*DmsClassData);*
SAP function module:	CAD_DISPLAY_BOM_WITH_SUB_ITEMS

See also:

Reading the BOM Item Table [Seite 583]

Reading the BOM Sub-Item Table [Extern]

Functions for Engineering Change Management

The CAD interface supports the following functions for processing change master records and ECRs:

CHGCR - Creating a Change Master Record [Seite 187]

CHGCH - Changing a Change Master Record [Seite 193]

CHGRQ - Displaying a Change Master Record [Seite 202]

ECRCR - Creating an ECR [Seite 211]

ECRCH - Changing an ECR [Seite 217]

ECRRQ - Displaying an ECR [Seite 227]

CHGCR - Creating a Change Master Record

The CAD interface allows you to create change master records in the SAP System from the external application program. You can maintain all change master record data, with the exception of long texts. When you create a new change master record, you can classify the change master record. You can also set the indicators that define how the change master record can be used in changes.

When you call the function, you send the change master record data to the SAP System as defined in table TCIU. The SAP System then sends the numbers of the change objects for which object indicators can be set in the next communication step *CHGCR OB*. If you want the change number to be used for an object type, send the relevant indicators for this object type to the SAP System.

The processing of further communication depends on whether you want to classify the change master record or not.

The following functions for creating change master records are supported:

- Creating a Change Master Record Without Classification [Seite 188]
- Creating a Change Master Record With Classification [Seite 190]

Creating a Change Master Record Without Classification

The table below describes the communication flow if you do not want to classify the change master record. Once the data of the change master record is sent to the SAP System, the change master record is created and the message output by the relevant transaction is returned to the external system.

Communication flow for function CHGCR MR without classification

SP code	Direction	Data transferred	Length
MR	→	Change master record data as defined in table TCIU	variable
OB	←	First object type &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Description of object type &)	variable
OB	→	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
MR	←	Function message	variable

Creating a Change Master Record With Classification

If you want to classify the change master record, the communication flow is more extensive, because information on characteristics also needs to be transferred.

If you want to classify the change master record, you must send the *class type* and *class* you require with the change master record data. You can also specify the scope to be used. The scope acts as a filter for selecting the relevant characteristics of the class. The scope selects according to whether characteristics are required or optional, whether they have values assigned or not, and whether they are inherited.

In a further communication step, the SAP interface program then sends the characteristics selected by the scope defined to the external application program, together with any default values. In the external system, values are assigned to the characteristics and transferred to the SAP System. You can use the help functions *HLPRQ SC* and *HLPRQ SE* to display allowed values. The change master record is created in the SAP System with its classification data.

Communication flow for function CHGCR MR with classification

SP code	Direction	Data transferred	Length
MR	→	Change master record data as defined in table TCIU including class type, class, and scope	variable
OB	←	First object type &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Description of object type &)	variable

OB	→	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
RF	→	Class type &)	variable
		Class &)	variable
		Indicator: required characteristics &)	variable
		Indicator: optional characteristics &)	variable
		Indicator: characteristics with assigned values &)	variable
		Indicator: characteristics without assigned values &)	variable
		Indicator: inherited characteristics &)	variable
		Indicator: standard class &)	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Assigned Characteristic Values [Seite 120]	variable
MR	←	Function message	variable

After communication step *CHGCR CR* (SAP→CAD), you can either assign values to the characteristics or continue with a different combination of class type, class, and scope.

When you create a new change master record, you can classify the change master record in a maximum of one class. You cannot classify the document in several classes at once in this way.

Creating a Change Master Record With Classification

However, you can perform multiple classification with the *'Change change master record with classification' CHGCH MC* function.

In communication step *CHGCR CR* (CAD→SAP), the characteristic values assigned in the external system are sent to the SAP System. Please note that a characteristic may occur more than once in this list. This is the case if multiple values are assigned to a characteristic.

If no data on characteristic values is sent to the SAP System in this communication step - in other words, if the send string has data length '0', the change master record is only allocated to the class. No characteristic values are assigned in this case.

In the customizing functions for the CAD interface, you can define whether document numbers can be assigned externally only, internally only, or both.

This function is called from the external application program as follows:

hd = SapChgcr(scode, sndstr, rcvstr, msg).

CHGCH - Changing a Change Master Record

The following functions for changing change master records are supported:

- Changing a Change Master Record Without Classification [Seite 194]
- Changing a Change Master Record With Classification [Seite 196]

Changing a Change Master Record Without Classification

This function is identified by sub-process code *MR*.

The function allows you to change data in a change master record, with the exception of classification data. The communication flow of function *CHGCH MR* is described in the table below.

Communication flow for function CHGCH MR

SP code	Direction	Data transferred	Length
MR	→	Key data of change master record as defined in table TCIU	variable
MD	←	Material master data in SAP System as defined in table TCID	variable
OB	←	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		Description of object type &)	variable
MD	→	Changed change master record data as defined in table TCIU	variable

OL	←	First object type &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Description of object type &)	variable
OL	→	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
MR	←	Function message	variable

When you call the function, the number of the change master record is sent to the SAP System. The SAP interface program then sends the existing change master record data as defined in settings in customizing to the external application program. At the same time, a lock entry is created for change master record in order to prevent concurrent access by other users.

The changes made to the change master record data in the external system are returned to the SAP System in a further communication step. The SAP interface program then triggers the changes in the SAP System.

Changing a Change Master Record with Classification

This function is identified by sub-process code *MC*.

The function allows you to change data in the change master record as in function *CHGCH MR*. You can also maintain classification data. You have the following options:

- changing an existing allocation and/or existing characteristic values
- creating an allocation to a new class
- deleting the allocation to a class

You can only create, change, or delete the allocation to one class in one function call. You cannot process several class allocations in one function.

Please note that classification is connected to engineering change management. This means that you can make all changes to classification data with history using either a date or a change number.

The communication flow for function *CHGCH MC* is the same as for function *CHGCH MR* up to communication step *CHGCH OB* (SAP-->CAD). When the changed change master record data is sent to the SAP System in the following communication step *CHGCH MD* (CAD→SAP), you can also send values for the *class type* and *class* whose allocation you want to change.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes to which the change master record is allocated is sent to the external application program.
  - If you enter a value for the class type, only the classes of this class type are sent to the external application program.
  - There is also a further restriction you can make regarding the standard class. If a standard class is defined for the change master record, or if the change master record is only allocated to one class, the communication step for transferring a list of classes is no longer necessary. In this case, the information on characteristics and assigned values is sent directly to the external program.

The communication flow of function *CHGCH MC* when changing an existing allocation is described in the table below.

Communication flow for function CHGCH MC for changing an existing allocation

SP code	Direction	Data transferred	Length
MC	→	Key data of change master record as defined in table TCIU	variable
MD	←	Material master data in SAP System as defined in table TCID	variable

OB	←	First object type &) Object type Indicator: object type active for change number &) Indicator: object type locked for changes &) Indicator: Display part objects for object &) Indicator: Management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Description of object type &) …. . Last object type &) Object type Indicator: object type active for change number &) Indicator: object type locked for changes &) Indicator: Display part objects for object &) Indicator: Management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Description of object type &)	variable variable variable variable variable variable variable variable variable variable . . variable variable variable variable variable variable variable variable variable variable
MD	→	Changed change master record data as defined in table TCIU including class type, class, and scope	variable
OL	←	First object type &) Description of object type &) . . . Last object type &) Description of object type &)	variable variable . . . variable variable
OL	→	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &)	variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable
RC	←	List of All Classes [Seite 126], in which the change master record is classified	variable

Changing a Change Master Record with Classification

RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

After communication step *CHGCH CR* (SAP-->CAD), you can either assign values to the characteristics or continue with a different combination of class type, class, and scope.

Communication step *CHGCH RC* (SAP→CAD) only applies if the restrictions sent in communication step *CHGCH MD* (CAD→SAP) produced a list of several classes in which the change master record is classified. The structure of the data string containing information on the classes in which the change master record is classified is shown in List of All Classes [Seite 126].

If your change to classification data is an allocation to an additional class, the communication flow is as follows:

Communication flow for function CHGCH MC for creating an allocation to an additional class

SP code	Direction	Data transferred	Length
MC	→	Key data of change master record as defined in table TCIU	variable
MD	←	Material master data in SAP System as defined in table TCID	variable
OB	←	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &)	variable variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable variable

MD	→	Changed change master record data as defined in table TCIU including class type, class, and scope	variable
OL	←	First object type &) Description of object type &) . . . Last object type &) Description of object type &)	variable variable . . . variable variable
OL	→	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &)	variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable
RC	←	List of All Classes [Seite 126], in which the change master record is classified	variable
NC	→	New class type &) New class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←←	Function message	variable

If you want to delete an existing allocation to a class, the communication flow is as follows:

Communication flow for function CHGCH MC for deleting an allocation to a class

SP code	Direction	Data transferred	Length
MC	→	Key data of change master record as defined in table TCIU	variable
MD	←	Material master data in SAP System as defined in table TCID	variable
OB	←	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &)	variable variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable variable
MD	→	Changed change master record data as defined in table TCIU including class type, class, and scope	variable
OL	←	First object type &) Description of object type &) . . . Last object type &) Description of object type &)	variable variable . . . variable variable

OL	→	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &)	variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable
RC	←	List of All Classes [Seite 126], in which the change master record is classified	variable
DC	→	Class type to be deleted &) Class to be deleted &)	variable variable
MC	←	Function message	variable

This function is called from the external application program as follows:

hd = SapChgch(scode, sndstr, rcvstr, msg).

CHGRQ - Displaying a Change Master Record

The following functions for displaying change master records are supported:

- Displaying a Change Master Record Without Classification [Seite 203]
- Displaying a Change Master Record With Classification [Seite 205]
- Finding a Change Master Record via Matchcode [Seite 208]

Displaying a Change Master Record Without Classification

This function is identified by sub-process code *MR*.

It allows you to display data in a change master record, with the exception of classification data. The communication flow of function *CHGRQ MR* is described in the table below.

Communication flow for function CHGRQ MR

SP code	Direction	Data transferred	Length
MR	→	Key data of change master record as defined in table TCIU	variable
MR	←	Material master data in SAP System as defined in table TCID	variable
OB	←	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		Description of object type &)	variable

When you call the function, the number of the change master record is sent to the SAP System. The SAP interface program then sends the existing change master record data as defined in settings in Customizing to the external application program. The object indicators set in the change master record are also sent in a separate record.

Displaying a Change Master Record With Classification

This function is identified by sub-process code *MC*.

It allows you to display data in a change master record as in function *CHGRQ MR.* You can also display classification data. The communication flow of function *CHGRQ MC* is described in the table below.

Communication flow for function CHGRQ MC

SP code	Direction	Data transferred	Length
MC	→	Key data of change master record as defined in table TCIU including class type, class, and scope	variable
MC	←	Material master data in SAP System as defined in table TCID	variable
OB	←	First object type &) Object type Indicator: object type active for change number &) Indicator: object type locked for changes &) Indicator: Display part objects for object &) Indicator: Management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Description of object type &) . . Last object type &) Object type Indicator: object type active for change number &) Indicator: object type locked for changes &) Indicator: Display part objects for object &) Indicator: Management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Description of object type &)	variable variable variable variable variable variable variable variable variable variable … variable variable variable variable variable variable variable variable variable variable
RC	←	List of All Classes [Seite 126], in which the change master record is classified	variable
RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable

RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable

When you call function *CHGRQ MC*, you can send values for the *class type* and *class* whose allocation you want to display.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes to which the change master record is allocated is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class.

  In this case, the information on characteristics and assigned values is sent directly to the external program.

  – You enter the class that the allocation is to be displayed for.

  – The restrictions in force when you call the function apply to only one class

  – The change master record is only classified in one class.

- You can also enter a date or a change number. Since classification is made with history, the status on a specific date is displayed. If you do not enter a date, the current date is taken as default.

  – If the restrictions on classification that you send with the function call are such that **several** classes are selected, this list of classes is sent to the external application program in communication step *CHGRQ RC* (SAP→CAD). In the external system, you select the class you are interested in and return the selected class to the SAP System in communication step *CHGRQ RC* (CAD→SAP).

    You then see the characteristic and characteristic values for the class and scope you require. As shown in the table above, communication step *CHGRQ RC* (CAD-->SAP) can be repeated as often as you require. In this way, you can display the classification data of the change master record in all the classes in the previous list. You can also vary the scope. This means that you can only terminate function *CHGRQ MC* with function *BREAK*, because the function does not come to a natural end.

  – If only **one** class is selected, or if the change master record is classified in exactly one class, you can only vary the scope in communication step *CHGRQ RC* (CAD→SAP).

  Please note that you can only use function *CHGRQ RC* (CAD-->SAP) to display classification data of the classes selected when you called the function. Classification data for classes that were not selected cannot be displayed.

This means that if the change master record is classified in class types ‘001’ and ‘002’, and you restrict the class type to class type ‘001’ when you call function *CHGRQ MC* (CAD→SAP), the classes of class type ‘002’ are not displayed. You cannot use function *CHGRQ RC* (CAD→SAP) and enter class type ‘002’ to display these classes. You need to call function *CHGRQ MC* again.

Finding a Change Master Record via Matchcode

This function is identified by sub-process code *RL*.

It allows you to find a change master record in the SAP System if you do not know the change number but do know other master data, such as the description or the valid-from date. You can also use the find objects function in the classification system to find change master records. This function is described in the section on the classification system.

The matchcode functionality allows you to define a matchcode ID such that it starts the function for finding objects via classes or a report, instead of the standard matchcode search function. These matchcode IDs cannot be processed using the CAD interface. If you select one of these matchcode IDs, you see an error message telling you that the matchcode ID cannot be used.

When you use a matchcode, either:

- You do not know the field structure of the selected matchcode ID, or
- You know the field structure of the selected matchcode ID.

The communication flow of function *CHGRQ RL* is described in the tables below.

Communication flow for function CHGRQ RL if you do not know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &)	variable
		Matchcode sub-ID &)	variable
		Description of matchcode ID &)	variable
		.	.
		.	.
		.	.
		Last matchcode ID &)	variable
		Matchcode sub-ID &)	variable
		Description of matchcode ID &)	variable
ID	→	Selected matchcode ID &)	variable
		Matchcode sub-ID &)	variable

RF	←	First matchcode field &) Length of first matchcode field &) . . . Last matchcode field &) Length of last matchcode field &)	variable variable . . . variable variable
RF	→	Matchcode selection string	variable
CL	←	Material master data in SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Material master data in SAP System as defined in table TCID	variable

Communication flow for function CHGRQ RL if you know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Matchcode selection string	variable
CL	←	Material master data in SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Material master data in SAP System as defined in table TCID	variable

Finding a Change Master Record via Matchcode

The matchcode sub-ID is still included in the transferred data strings to ensure compatibility with R/2. In data strings sent from the R/3 System to the external application program, the matchcode sub-ID is SPACE. In data strings sent from the external application program to the SAP System, you can either send the additional separator character or not.

- After you call function *CHGRQ RL,* first the matchcode IDs defined in the SAP *Engineering Change Management* system for change master records are sent to the external application program. For further processing, there is a difference depending on whether you know the field structure of the selected matchcode or not.
  - If you do **not** know the field structure, you send the selected matchcode ID to the SAP System in the next communication step in order to get the field structure.
  - However, if you know the field structure, you can use the next communication step to send the matchcode selection string. To do this, you use the matchcode identifier (usually the '0' character). The matchcode identifier is transferred to the external application program using system function *SYSRQ PA*.
- When you use matchcodes to find change master records, please note that the master record of each change master record found by the matchcode search is read in order to procure the data that is not stored in matchcode records. However, this may lead to runtime problems if the matchcode selection criteria are not specific enough and therefore a large number of change master records is selected. For this reason, control table TCIS contains parameter TCIS-MCMAX, which defines the maximum number of matchcode records, after which communication with the external application program is triggered. If such a communication is necessary, you use sub-process code *CL*. If you call function *ECRRQ CL,* you can then request the following matchcode records.

As of release 3.0, all matchcode selection functions were changed such that functions *CHGRQ ID* and *CHGRQ RF* can be called in any communication step. This has the advantage that you do not need to start again by calling function *CHGRQ RL* if your matchcode search is unsuccessful. Instead, you can enter new values in the matchcode fields directly if you want to use the current matchcode ID, or switch to another matchcode ID. However, this means that the matchcode selection functions no longer come to a natural end. As of release 3.0, you terminate these functions using the *BREAK* function

This function is called from the external application program as follows:

hd = SapChgrq(scode, sndstr, rcvstr, msg).

ECRCR - Creating an ECR

The CAD interface allows you to create ECRs in the SAP System from your external application program. You can maintain all data in the ECR with the exception of long texts. When you create a new ECR, you can maintain its classification data and the indicators that control its use in changes.

When you call the function, you must send the change request data to the SAP System as defined in table TCIU. The SAP System then sends the numbers of the change objects for which object indicators can be set in the next communication step *CHGCR OB*. If you want the change request to be used for an object type, send the relevant indicators for this object type to the SAP System.

The processing of further communication depends on whether you want to classify the change request or not.

The following functions for creating change requests are supported:

- Creating an ECR Without Classification [Seite 212]
- Creating an ECR With Classification [Seite 214]

Creating an ECR Without Classification

The table below describes the communication flow if you do not want to classify the ECR. Once the data of the ECR is sent to the SAP System, the ECR is created and the message output by the relevant transaction is returned to the external system.

Communication flow for function ECRCR MR without classification

SP code	Direction	Data transferred	Length
MR	→	ECR data as defined in table TCIU	variable
OB	←	First object type &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Description of object type &)	variable
OB	→	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
MR	←	Function message	variable

Creating an ECR With Classification

If you want to classify the ECR, the communication flow is more extensive, because data on characteristics has to be transferred.

- If you want to classify the ECR, you must send the *class type* and *class* you require with the change request data.

  You can also specify the scope to be used. The scope acts as a filter for selecting the relevant characteristics of the class. The scope selects according to whether characteristics are required or optional, whether they have values assigned or not, and whether they are inherited.

- In a further communication step, the SAP interface program then sends the characteristics selected by the scope defined to the external application program, together with any default values. In the external system, values are assigned to the characteristics and transferred to the SAP System. You can use the help functions *HLPRQ SC* and *HLPRQ SE* to display allowed values. The ECR is created in the SAP System with its classification data.

Communication flow for function ECRCR MR with classification

SP code	Direction	Data transferred	Length
MR	→	ECR data as defined in table TCIU including class type, class, and scope	variable
OB	←	First object type &) Description of object type &) . . . Last object type &) Description of object type &)	variable variable . . . variable variable

OB	→	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
RF	→	Class type &)	variable
		Class &)	variable
		Indicator: required characteristics &)	variable
		Indicator: optional characteristics &)	variable
		Indicator: characteristics with assigned values &)	variable
		Indicator: characteristics without assigned values &)	variable
		Indicator: inherited characteristics &)	variable
		Indicator: standard class &)	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Assigned Characteristic Values [Seite 120]	variable
MR	←	Function message	variable

After communication step *ECRCR CR* (SAP→CAD), you can either assign values to the characteristics or continue with a different combination of class type, class, and scope.

Creating an ECR With Classification

- When you create a new ECR, you can only classify the ECR in one class. You cannot classify the document in several classes at once in this way. However, you can perform a multiple classification using the *'Change ECR with classification' ECRCH MC* function.
  - In communication step *ECRCR CR* (CAD→SAP), the characteristic values assigned in the external system are sent to the SAP System. Please note that a characteristic may occur more than once in this list. This is the case if multiple values are assigned to a characteristic.
  - If no data on characteristic values is sent to the SAP System in this communication step - in other words, if the send string has data length '0' - the ECR is allocated to the class only. No characteristic values are assigned in this case.
- In the customizing functions for the CAD interface, you can define whether document numbers can be assigned externally only, internally only, or both.

This function is called from the external application program as follows:

hd = SapEcrcr(scode, sndstr, rcvstr, msg).

ECRCH - Changing an ECR

The following functions for changing ECRs (engineering change requests) are supported:

- Changing an ECR Without Classification [Seite 218]
- Changing an ECR With Classification [Seite 220]

Changing an ECR Without Classification

This function is identified by sub-process code *MR*.

It allows you to change data in the ECR, with the exception of classification data. The communication flow of function *ECRCH MR* is described in the table below.

Communication flow for function ECRCH MR

SP code	Direction	Data transferred	Length
MR	→	Key data of change request as defined in table TCIU	variable
MD	←	ECR data in the SAP System as defined in table TCID	variable
OB	←	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		Description of object type &)	variable
MD	→	Changed data of ECR as defined in table TCIU	variable

OL	←	First object type &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Description of object type &)	variable
OL	→	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
MR	←	Function message	variable

When you call the function, the key data of the ECR is sent to the SAP System. The SAP interface program then sends existing ECR data as defined in settings in Customizing to the external application program. At the same time, a lock entry is created for the ECR, in order to prevent concurrent access by other users.

The changes made to the ECR data in the external system are returned to the SAP System in a further communication step. The SAP interface program then triggers the changes in the SAP System.

Changing an ECR with Classification

This function is identified by sub-process code *MC*.

The function allows you to change data in the ECR as in function *ECRCH MR.* You can also maintain classification data. You have the following options:

- changing an existing allocation and/or existing characteristic values
- creating an allocation to a new class
- deleting the allocation to a class

You can only create, change, or delete the allocation to one class in one function call. You cannot process several class allocations in one function.

Please note that classification is connected to engineering change management. This means that you can make all changes to classification data with history using either a date or a change number.

The communication flow for function *ECRCH MC* is the same as for function *ECRCH MR* up to communication step *ECRCH OB* (SAP-->CAD).

- When the changed ECR data is sent to the SAP System in the following communication step *ECRCH MD* (CAD-->SAP), you can also send values for the *class type* and *class* whose allocations you want to change. This data has a restrictive function.
  - If you do not enter these values, a data string including all classes in which the ECR is classified is sent to the external application program.
  - If you enter a value for the class type, only the classes of this class type are sent to the external application program. There is also a further restriction you can make regarding the standard class.
- If a standard class is defined for the change master record, or if the change master record is only allocated to one class, the communication step for transferring a list of classes is no longer necessary. In this case, the information on characteristics and assigned values is sent directly to the external program.

Communication flow when changing the allocation to a class

The communication flow of function *ECRCH MC* when changing an existing class allocation is described in the table below.

Communication flow for function ECRCH MC for changing an existing class allocation

SP code	Direction	Data transferred	Length
MC	→	Key data of change request as defined in table TCIU	variable
MD	←	ECR data in the SAP System as defined in table TCID	variable

OB	←	First object type &) Object type Indicator: object type active for change number &) Indicator: object type locked for changes &) Indicator: Display part objects for object &) Indicator: Management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Description of object type &) . . Last object type &) Object type Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &)	variable variable variable variable variable variable variable variable variable variable . variable variable variable variable variable variable variable variable variable
MD	→	Changed data of ECR as defined in table TCIU including class type, class, and scope	variable
OL	←	First object type &) Description of object type &) . . . Last object type &) Description of object type &)	variable variable . . . variable variable
OL	→	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &)	variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable
RC	←	List of All Classes [Seite 126] in which the change request is classified	variable

Changing an ECR with Classification

RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

After communication step *ECRCH CR* (SAP→CAD), you can either assign values to the characteristics or continue with a different combination of class type, class, and scope.

Communication step *ECRCH RC* (SAP-->CAD) only applies if the restrictions sent in communication step *ECRCH MD* (CAD-->SAP) produced a list of several classes in which the ECR is classified.

Communication flow when creating an additional allocation to a class

If your change to classification data is an allocation to an additional class, the communication flow is as follows:

Communication flow for function ECRCH MC for creating an allocation to an additional class

SP code	Direction	Data transferred	Length
MC	→	Key data of change request as defined in table TCIU	variable
MD	←	ECR data in the SAP System as defined in table TCID	variable

OB	←	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &)	variable variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable variable
MD	→	Changed data of ECR as defined in table TCIU including class type, class, and scope	variable
OL	←	First object type &) Description of object type &) . . . Last object type &) Description of object type &)	variable variable . . . variable variable
OL	→	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &)	variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable
RC	←	List of All Classes [Seite 126] in which the change request is classified	variable

Changing an ECR with Classification

NC	→	New class type &) New class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

Communication flow when deleting an allocation to a class

If you want to delete an existing allocation to a class, the communication flow is as follows:

Communication flow for function ECRCH MC for deleting an existing class allocation

SP code	Direction	Data transferred	Length
MC	→	Key data of change request as defined in table TCIU	variable
MD	←	ECR data in the SAP System as defined in table TCID	variable

OB	←	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) Description of object type &)	variable variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable variable
MD	→	Changed data of ECR as defined in table TCIU including class type, class, and scope	variable
OL	←	First object type &) Description of object type &) . . . Last object type &) Description of object type &)	variable variable . . . variable variable
OL	→	First object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &) . . . Last object type &) Indicator: object type active for change number &) Indicator: management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Indicator: object type locked for changes &)	variable variable variable variable variable variable variable . . . variable variable variable variable variable variable variable
RC	←	List of All Classes [Seite 126] in which the change request is classified	variable
DC	→	Class type to be deleted &) Class to be deleted &)	variable variable
MC	←	Function message	variable

This function is called from the external application program as follows:

Changing an ECR with Classification

hd = SapEcrch(scode, sndstr, rcvstr, msg).

ECRRQ - Displaying an ECR

The following functions for displaying ECRs are supported:

- Displaying an ECR Without Classification [Seite 228]
- Displaying an ECR With Classification [Seite 230]
- Finding an ECR via Matchcode [Seite 233]

Displaying an ECR Without Classification

This function is identified by sub-process code *MR*.

It allows you to display the data in an ECR, with the exception of classification data. The communication flow of function *ECRRQ MR* is described in the table below.

Communication flow for function ECRRQ MR

SP code	Direction	Data transferred	Length
MR	→	Key data of change request as defined in table TCIU	variable
MR	←	ECR data in the SAP System as defined in table TCID	variable
OB	←	First object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		Description of object type &)	variable
		.	.
		.	.
		.	.
		Last object type &)	variable
		Indicator: object type active for change number &)	variable
		Indicator: management record required for each object &)	variable
		Indicator: object management record generated &)	variable
		Indicator: generation only for new records &)	variable
		Indicator: generation in dialog mode &)	variable
		Indicator: object type locked for changes &)	variable
		Description of object type &)	variable

When you call the function, the key fields of the ECR are sent to the SAP System. The SAP interface program then sends existing ECR data as defined in settings in customizing to the external application program. The object indicators set in the ECR are sent in a subsequent record.

Displaying an ECR With Classification

This function is identified by sub-process code *MC*.

It allows you to display the data in an ECR as in function *ECRRQ MR*. You can also display classification data. The communication flow of function *ECRRQ MC* is described in the table below.

Communication flow for function ECRRQ MC

SP code	Direction	Data transferred	Length
MC	→	Key data of change request as defined in table TCIU including class type, class, and scope	variable
MC	←	ECR data in the SAP System as defined in table TCID	variable
OB	←	First object type &) Object type Indicator: object type active for change number &) Indicator: object type locked for changes &) Indicator: Display part objects for object &) Indicator: Management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Description of object type &) . . Last object type &) Object type Indicator: object type active for change number &) Indicator: object type locked for changes &) Indicator: Display part objects for object &) Indicator: Management record required for each object &) Indicator: object management record generated &) Indicator: generation only for new records &) Indicator: generation in dialog mode &) Description of object type &)	variable variable variable variable variable variable variable variable variable variable . … variable variable variable variable variable variable variable variable variable variable
RC	←	List of All Classes [Seite 126] in which the change request is classified	variable
RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable

RC	→	Class type &)	variable
		Class &)	variable
		Indicator: required characteristics &)	variable
		Indicator: optional characteristics &)	variable
		Indicator: characteristics with assigned values &)	variable
		Indicator: characteristics without assigned values &)	variable
		Indicator: inherited characteristics &)	variable
		Indicator: standard class &)	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
Check out files	←	Available Values for Characteristics [Seite 119]	variable

When you call function *ECRRQ MC,* you can send values for the *class type* and *class* whose allocation you want to display.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes in which the ECR is classified is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class.

  In this case, the information on characteristics and assigned values is sent directly to the external program.

  – The restrictions in force when you call the function apply to only one class

  – The change request is only classified in one class.

- You can also enter a date or a change number. Since classification is made with history, the status on a specific date is displayed. If you do not enter a date, the current date is taken as default.

  – If the restrictions on classification that you send with the function call are such that **several** classes are selected, this list of classes is first sent to the external application program in communication step *ECRRQ RC* (SAP-->CAD). In the external system, you select the class you are interested in and return the selected class to the SAP System in communication step *ECRRQ RC* (CAD-->SAP).

    You then see the characteristic and characteristic values for the class and scope you require. As shown in the table above, communication step *ECRRQ RC* (CAD-->SAP) can be repeated as often as you require. In this way, you can display the classification data of the ECR in all the classes in the previous list. You can also vary the scope. This means that you can only terminate function *ECRRQ MC* with the *BREAK* function because the function does not come to a natural end.

  – If only **one** class is selected, or if the ECR is classified in exactly one class, you can only vary the scope in communication step *ECRRQ RC* (CAD-->SAP).

Please note that you can only use function *ECRRQ RC* (CAD→SAP) to display classification data of the classes selected when you called the function. Classification data for classes that were not selected cannot be displayed. This means that if the change request is classified in class types ‘001’ and ‘002’, and you restrict the class type to class type ‘001’ when you call function *CHGRQ MC* (CAD→SAP), the classes of class type ‘002’ are not displayed. You cannot use function *ECRRQ RC* (CAD→SAP) and enter class type ‘002’ to display these classes. You need to call function *ECRRQ MC* again.

Finding an ECR via Matchcode

This function is identified by sub-process code *RL*.

It allows you to find an ECR in the SAP System if you do not know the change number, but do know other master data, such as the description or valid-from date. You can also use the find objects function in the classification system to find ECRs. This function is described in the section on the classification system.

The matchcode functionality allows you to define a matchcode ID such that it starts the function for finding objects via classes or a report, instead of the standard matchcode search function. These matchcode IDs cannot be processed using the CAD interface. If you select one of these matchcode IDs, you see an error message telling you that the matchcode ID cannot be used.

When you use a matchcode, either:

- You do not know the field structure of the selected matchcode ID, or
- You know the field structure of the selected matchcode ID.

The communication flow of function *ECRRQ RL* is described in the tables below.

Communication flow for function ECRRQ RL if you do not know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &)	variable
		Matchcode sub-ID &)	variable
		Description of matchcode ID &)	variable
		.	.
		.	.
		.	.
		Last matchcode ID &)	variable
		Matchcode sub-ID &)	variable
		Description of matchcode ID &)	variable
ID	→	Selected matchcode ID &)	variable
		Matchcode sub-ID &)	variable

Finding an ECR via Matchcode

RF	←	First matchcode field &) Length of first matchcode field &) . . . Last matchcode field &) Length of last matchcode field &)	variable variable . . . variable variable
RF	→	Matchcode selection string	variable
CL	←	ECR data in the SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	ECR data in the SAP System as defined in table TCID	variable

Communication flow for function ECRRQ RL if you know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Matchcode selection string	variable
CL	←	ECR data in the SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	ECR data in the SAP System as defined in table TCID	variable

The matchcode sub-ID is still included in the transferred data strings to ensure compatibility with R/2. In data strings sent from the R/3 System to the external application program, the matchcode sub-ID is SPACE. In data strings sent from the external application program to the SAP System, you can either send the additional separator character or not.

- After you call function *ECRRQ RL*, first the matchcode IDs defined in the SAP *Engineering Change Management* system for ECRs are sent to the external application program. For further processing, there is a difference depending on whether you know the field structure of the selected matchcode or not.
  - If you do **not** know the field structure, you send the selected matchcode ID to the SAP System in the next communication step in order to get the field structure.
  - However, if you know the field structure, you can use the next communication step to send the matchcode selection string. To do this, you use the matchcode identifier (usually the '0' character). The matchcode identifier is transferred to the external application program using system function *SYSRQ PA*.
- When you use matchcodes to find ECRs, please note that the master record of each ECR found by the matchcode search is read in order to procure the data that is not stored in matchcode records. However, this may lead to runtime problems if the matchcode selection criteria are not specific enough and therefore a large number of ECRs is selected. For this reason, control table TCIS contains parameter TCIS-MCMAX, which defines the maximum number of matchcode records, after which communication with the external application program is triggered. If such a communication is necessary, you use sub-process code *CL*. If you call function *ECRRQ CL,* you can then request the following matchcode records.

As of release 3.0, all matchcode selection functions were changed such that functions *ECRRQ ID* and *ECRRQ RF* can be called in any communication step. This has the advantage that you do not need to start again by calling function *ECRRQ RL* if your matchcode search is unsuccessful. Instead, you can enter new values in the matchcode fields directly if you want to use the current matchcode ID, or switch to another matchcode ID. However, this means that the matchcode selection functions no longer come to a natural end. As of release 3.0, you terminate these functions using the *BREAK* function

This function is called from the external application program as follows:

hd = SapEcrrq(scode, sndstr, rcvstr, msg).

Functions for Plant Maintenance

The CAD interface supports the following functions for processing plant maintenance objects:

EQICR - Creating an Equipment Master Record [Seite 237]
EQICH - Changing an Equipment Master Record [Seite 241]
EQIRQ - Displaying an Equipment Master Record [Seite 247]
PMNCR - Creating a Maintenance Notification [Seite 255]
TPCCR - Creating a Functional Location [Seite 256]
TPCCH - Changing a Functional Location [Seite 260]
TPCRQ - Displaying a Functional Location [Seite 265]

EQICR - Creating an Equipment Master Record

The CAD interface allows you to create equipment master records in the SAP System from the external application program. You can maintain all equipment master data, with the exception of long texts. When you create a new equipment master record, you can install this equipment at a functional location immediately.

When you call the function, the equipment master record data is sent to the SAP System. The structure for transfer is defined in table TCIU. The communication flow from here depends on whether you want to classify the equipment master record or not.

The following functions for creating equipment master records are supported:

- Creating an Equipment Master Record Without Classification [Seite 238]
- Creating an Equipment Master Record With Classification [Seite 239]

Creating an Equipment Master Record Without Classification

The table below shows the communication flow that applies if you do not want to classify the equipment master record. Once the equipment data has been sent to the SAP System, the equipment master record is created, and the message output by the transaction concerned is returned to the external system.

Communication flow for function EQICR MR without classification

SP code	Direction	Data transferred	Length
MR	→	Equipment master data as defined in table TCIU	variable
MR	←	Function message	variable

Creating an Equipment Master Record With Classification

If you want to classify the equipment, the communication flow is more extensive, because characteristics data has to be transferred as well.

- If you want to classify the equipment, you must send the *class type* and *class* you require with the equipment data.

  You can also specify the scope to be used. The scope acts as a filter for selecting the relevant characteristics of the class. The scope selects according to whether characteristics are required or optional, whether they have values assigned or not, and whether they are inherited.

- In a further communication step, the SAP interface program then sends the characteristics selected by the scope defined to the external application program, together with any default values.

- In the external system, values are assigned to the characteristics and transferred to the SAP System. You can use the help functions *HLPRQ SC* and *HLPRQ SE* to display allowed values. The equipment master record is then created in the SAP System with its classification data.

Communication flow for function EQICR MR with classification

SP code	Direction	Data transferred	Length
MR	→	Equipment data as defined in table TCIU including class type, class, and scope	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Assigned Characteristic Values [Seite 120]	variable
MR	←	Function message	variable

After communication step *EQICR CR* (SAP→CAD), you can either assign values to the characteristics or continue with a changed combination of class type, class, and scope.

- When you create a new equipment master record, you can classify the equipment in no more than one class. You cannot classify the document in several classes at once in this way. However, you can do a multiple classification using function *Change equipment master with classification (EQICH MC)*.

Creating an Equipment Master Record With Classification

- In communication step *EQICR CR* (CAD→SAP) the characteristic values assigned in the external system are sent to the SAP System. A characteristic may appear more than once in this list. This is the case if multiple values are assigned to a characteristic.

  If no data on characteristic values is sent to the SAP System in this communication step - in other words, if the send string has data length ‘0’ - the ECR is allocated to the class only. No characteristic values are assigned in this case.

- In the customizing functions for the CAD interface, you can define whether document numbers can be assigned externally only, internally only, or both.

This function is called from the external application program as follows:

hd = SapEqicr(scode, sndstr, rcvstr, msg).

EQICH - Changing an Equipment Master Record

The following functions for changing equipment master records are supported:

- Changing an Equipment Master Record Without Classification [Seite 242]
- Changing an Equipment Master Record With Classification [Seite 243]
- Dismantle equipment master record [Seite 246]

Changing an Equipment Master Record Without Classification

This function is identified by sub-process code *MR*.

It allows you to change data in the equipment master record, with the exception of classification data. The communication flow of function *EQICH MR* is described in the table below.

Communication flow for function EQICH MR

SP code	Direction	Data transferred	Length
MR	→	Equipment key data as defined in table TCIU	variable
MD	←	Equipment data in the SAP System as defined in table TCID	variable
MD	→	Changed equipment data as defined in table TCIU	variable
MR	←	Function message	variable

When you call the function, the number of the equipment master record is sent to the SAP System. The SAP interface program then sends the equipment data as defined in settings in customizing to the external application program. At the same time, a lock entry is created for the equipment master record in order to prevent concurrent access by other users.

The changes made to the equipment data in the external system are returned to the SAP System in a further communication step. The SAP interface program then triggers the changes in the SAP System.

Changing an Equipment Master Record With Classification

This function is identified by sub-process code *MC*.

The function allows you to change data in the equipment master record as in function *EQICH MR*. You can also maintain classification data. You have the following options:

- changing an existing allocation and/or existing characteristic values
- creating an allocation to a new class
- deleting the allocation to a class

You can only create, change, or delete the allocation to one class in one function call. You cannot process several class allocations in one function.

Please note that classification is connected to engineering change management. This means that you can make all changes to classification data with history using either a date or a change number.

The communication flow for function *EQICH MC* is the same as for function *EQICH MR* up to communication step *EQICH MD* (SAP→CAD). When the changed equipment master data is sent to the SAP System in the following communication step *EQICH MD* (CAD→SAP), you can also send values for the *class type* and *class* whose allocation you want to change.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes in which the equipment is classified is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class. However, if a standard class is defined for the equipment, or if the equipment is only allocated to one class, the communication step for transferring a list of classes is no longer necessary. In this case, the information on characteristics and assigned values is sent directly to the external program.

The communication flow of function *EQICH MC* when changing an existing allocation is described in the table below.

Communication flow for function EQICH MC for changing an existing allocation

SP code	Direction	Data transferred	Length
MC	→	Equipment key data as defined in table TCIU	variable
MD	←	Equipment data in the SAP System as defined in table TCID	variable
MD	→	Changed equipment data as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], in which the equipment is classified	variable

Changing an Equipment Master Record With Classification

RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

After communication step *EQICH CR* (SAP→CAD), you can either assign characteristic values or continue with a different combination of class type, class, and scope.

Communication step *EQICH RC* (SAP→CAD) only takes place if the

restrictions made in communication step *EQICH MD* (CAD→SAP) result in more than one class that the equipment is classified in.

If your change to classification data is an allocation to an additional class, the communication flow is as follows:

Communication flow for function EQICH MC for creating an allocation to an additional class

SP code	Direction	Data transferred	Length
MC	→	Equipment key data as defined in table TCIU	variable
MD	←	Equipment data in the SAP System as defined in table TCID	variable
MD	→	Changed equipment data as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], in which the equipment is classified	variable
NC	→	New class type &) New class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable

RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

If you want to delete an existing allocation to a class, the communication flow is as follows:

Communication flow for function EQICH MC for deleting an existing allocation

SP code	Direction	Data transferred	Length
MC	→	Equipment key data as defined in table TCIU	variable
MD	←	Equipment data in the SAP System as defined in table TCID	variable
MD	→	Changed equipment data as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], in which the equipment is classified	variable
DC	→	Class type to be deleted &) Class to be deleted &)	variable variable
MC	←	Function message	variable

This function is called from the external application program as follows:

hd = SapEqich(scode, sndstr, rcvstr, msg).

Dismantling Equipment Master Records

This function is identified by sub-process code *DM*.

The function allows only the dismantling of equipment; no other data can be changed.

The communication flow of function *EQICH DM* is described in the table below.

Communication flow for function EQICH MR

SP code	Direction	Data transferred	Length
DM	–{}–→	Equipment number (CAD -> SAP)	variable
DM	←	Message (SAP -> CAD)	variable

EQIRQ - Displaying an Equipment Master Record

The following functions for displaying equipment master records are supported:

- Displaying an Equipment Master Record Without Classification [Seite 248]
- Displaying an Equipment Master Record With Classification [Seite 249]
- Finding an Equipment via Matchcode [Seite 251]
- Displaying an Equipment Hierarchy [Seite 254]

Displaying an Equipment Master Record Without Classification

This function is identified by sub-process code *MR*.

The function allows you to display data in the equipment master record, with the exception of classification data. The communication flow of function *EQIRQ MR* is described in the table below.

Communication flow for function EQIRQ MR

SP code	Direction	Data transferred	Length
MR	→	Equipment key data as defined in table TCIU	variable
MR	←	Equipment data in the SAP System as defined in table TCID	variable

When you call the function, the number of the equipment master record is sent to the SAP System. The SAP interface program then sends the equipment data as defined in settings in customizing to the external application program.

Displaying an Equipment Master Record With Classification

This function is identified by sub-process code *MC*.

It allows you to display data in the equipment master record as in function *EQIRQ MR.* You can also display classification data. The communication flow of function *EQIRQ MC* is described in the table below.

Communication flow for function EQIRQ MC

SP code	Direction	Data transferred	Length
MC	→	Equipment key data as defined in table TCIU including class type, class, and scope	variable
MC	←	Equipment data in the SAP System as defined in table TCID	variable
RC	←	List of All Classes [Seite 126], in which the equipment is classified	variable
RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable

When you call function *EQIRQ MC*, you can send values for the *class type* and *class* whose allocation you want to display immediately.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes in which the equipment is classified is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class.

Displaying an Equipment Master Record With Classification

In the following situations, the information on characteristics and assigned values is sent directly to the external program:

- You enter the class that the allocation is to be displayed for.
- The restrictions in force when you call the function apply to only one class
- The equipment is only allocated to one class.

- You can also enter a date or a change number. Since classification is made with history, the status on a specific date is displayed. If you do not enter a date, the current date is taken as a default.
  - If the restrictions on classification that you send with the function call are such that **several** classes are selected, this list of classes is first sent to the external application program in communication step *EQIRQ RC* (SAP→CAD). In the external system, you select the class you are interested in and return the selected class to the SAP System in communication step *EQIRQ RC* (CAD→SAP).

    You then see the characteristic and characteristic values for the class and scope you require. As shown in the table above, communication step *EQIRQ RC* (CAD→SAP) can be repeated as often as you require. In this way, you can display the classification data of the equipment in all the classes in the previous list. You can also vary the scope. This means that you can only terminate function *EQIRQ MC* with function *BREAK*, because the function does not come to a natural end.
  - If only **one** class is selected, or if the equipment is classified in exactly one class, you can only vary the scope in communication step *EQIRQ RC* (CAD→SAP).

  Please note that you can only use function *EQIRQ RC* (CAD→SAP) to display classification data of the classes selected when you called the function. Classification data for classes that were not selected cannot be displayed.

  This means that if the equipment is classified in class types ‘001’ and ‘002’, and you restrict the class type to class type ‘001’ when you call function *EQIRQ MC* (CAD→SAP), the classes of class type ‘002’ are not displayed. You cannot use function *EQIRQ RC* (CAD→SAP) and enter class type ‘002’ to display these classes. You need to call function *EQIRQ MC* again.

Finding an Equipment via Matchcode

This function is identified by sub-process code *RL*.

The function allows you to find an equipment in the SAP System if you do not know the equipment number, but do know other master data, such as the description or the cost center. You can also use the find objects function in the classification system to find equipments. This function is described in the section on the classification system.

The matchcode functionality allows you to define a matchcode ID such that it starts the function for finding objects via classes or a report, instead of the standard matchcode search function. These matchcode IDs cannot be processed using the CAD interface. If you select one of these matchcode IDs, you see an error message telling you that the matchcode ID cannot be used.

When you use a matchcode, either:

- You do not know the field structure of the selected matchcode ID, or
- You know the field structure of the selected matchcode ID.

The communication flow of function *EQIRQ RL* is described in the tables below.

Communication flow for function EQIRQ RL if you do not know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Selected matchcode ID &) Matchcode sub-ID &)	variable variable
RF	←	First matchcode field &) Length of first matchcode field &) . . . Last matchcode field &) Length of last matchcode field &)	variable variable . . . variable variable
RF	→	Matchcode selection string	variable
CL	←	Equipment data in the SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Equipment data in the SAP System as defined in table TCID	variable

Finding an Equipment via Matchcode

Communication flow for function EQIRQ RL if you know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Matchcode selection string	variable
CL	←	Equipment data in the SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Equipment data in the SAP System as defined in table TCID	variable

The matchcode sub-ID is still included in the transferred data strings to ensure compatibility with R/2. In data strings sent from the R/3 System to the external application program, the matchcode sub-ID is SPACE. In data strings sent from the external application program to the SAP System, you can either send the additional separator character or not.

- After you call function *EQIRQ RL,* first the matchcode IDs defined in the SAP plant maintenance system for equipments are sent to the external application program. For further processing, there is a difference depending on whether you know the field structure of the selected matchcode or not.
  - If you do **not** know the field structure, you send the selected matchcode ID to the SAP System in the next communication step in order to get the field structure.
  - However, if you know the field structure, you can use the next communication step to send the matchcode selection string. To do this, you use the matchcode identifier (usually the ‘0’ character). The matchcode identifier is transferred to the external application program using system function *SYSRQ PA*.
- When you use matchcodes to find equipments, please note that the master record of each equipment found by the matchcode search is read in order to procure the data that is not stored in matchcode records. However, this may lead to runtime problems if the matchcode selection criteria are not specific enough and therefore a large number of equipment master records is selected. For this reason, control table TCIS contains parameter TCIS-MCMAX, which defines the maximum number of matchcode records, after which communication with the external application program is triggered. If such a communication is necessary, you use sub-process code *CL*. If you call function *EQIRQ CL,* you can then request the following matchcode records.

As of release 3.0, all matchcode selection functions were changed such that functions *EQIRQ ID* and *EQIRQ RF* can be called in any communication step. This has the advantage that you do not need to start again by calling function *EQIRQ RL* if your matchcode search is unsuccessful. Instead, you can enter new values in the matchcode fields directly if you want to use the current matchcode ID, or switch to another matchcode ID. However, this means that the matchcode

selection functions no longer come to a natural end. As of release 3.0, you terminate these functions using the *BREAK* function

Displaying an Equipment Hierarchy

This function is identified by sub-process code *RH*.

It allows you to explode the hierarchy that contains the current equipment. You can set up an equipment hierarchy using functions *EQICR MR, EQICH MC,* or *EQICH MR* by entering a value for the superior equipment. You can restrict explosion of the hierarchy to one direction or other. You can also define the number of levels of the hierarchy you want to explode.

The communication flow of function *EQIRQ RH* is described in the table below.

Communication flow for function EQIRQ RH

SP code	Direction	Data transferred	Length
RH	→	Equipment number &) Effectivity date &) Number of explosion levels upward &) Number of explosion levels downward &)	variable variable variable variable
RH	←	Equipment data in the SAP System as defined in table TCID for each equipment found in the hierarchy	variable

When you call the function, you must send the equipment number and the effectivity date for which you want to explode the hierarchy. If you do not enter a date, the current date is used as a default. If you do not enter the number of explosion levels downward, the number is set to ‘99’. However, if you do not enter the number of explosion levels upward, the number is set to ‘0’. This means that the hierarchy above the current equipment is not exploded.

This function is called from the external application program as follows:

hd = SapEqirq(scode, sndstr, rcvstr, msg).

PMNCR - Creating a Maintenance Notification

The CAD interface allows you to create a maintenance notification in the SAP System from the external application program.

When you call the function, you send the maintenance notification data as defined in table TCIU to the SAP System. The maintenance notification is created in the SAP System, and a message is sent to the external application program. The communication flow for function *PMNCR MR* described in the table below.

Communication flow for function PMNCR MR

SP code	Direction	Data transferred	Length
MR	→	Maintenance notification data as defined in table TCIU	variable
MR	←	Function message	variable

This function is called from the external application program as follows:

hd = SapPmncr(scode, sndstr, rcvstr, msg).

TPCCR - Creating a Functional Location

The CAD interface allows you to create a functional location in the SAP System from the external application program. You can maintain all functional location data, with the exception of long texts.

When you call the function, you send the functional location data as defined in table TCIU to the SAP System. The processing of further communication depends on whether you want to classify the functional location or not.

The following functions for creating functional locations are supported:

- Creating a Functional Location Without Classification [Seite 257]
- Creating a Functional Location With Classification [Seite 258]

Creating a Functional Location Without Classification

The table below shows the communication flow if you do not want to classify the function location. Once the functional location data is sent to the SAP System, the master record for the functional location is created and a message is returned to the external system.

Communication flow for function TPCCR MR without classification

SP code	Direction	Data transferred	Length
MR	→	Functional location data as defined in table TCIU	variable
MR	←	Function message	variable

Creating a Functional Location With Classification

If you want to classify the functional location, the communication flow is more extensive in order to allow the transfer of characteristics data.

- If you want to classify the functional location, you must send the *class type* and *class* you require together with the data of the functional location. You can also specify the scope to be used. The scope acts as a filter for selecting the relevant characteristics of the class. The scope selects according to whether characteristics are required or optional, whether they have values assigned or not, and whether they are inherited.
- The SAP interface program then sends the characteristics selected according to the scope, and any default values, to the external application program in a further communication step. In the external system, values are assigned to the characteristics and transferred to the SAP System. You can use the help functions *HLPRQ SC* and *HLPRQ SE* to display allowed values. In the SAP System, the document info record is created with its classification data.

Communication flow for function TPCCR MR with classification

SP code	Direction	Data transferred	Length
MR	→	Functional location data as defined in table TCIU including class type, class, and scope	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Assigned Characteristic Values [Seite 120]	variable

After communication step *TPCCR CR* (SAP→CAD), you can either assign values to the characteristics or continue with a different combination of class type, class, and scope.

When you create a new functional location, you can only classify the functional location in one class. You cannot classify the document in several classes at once in this way. However, you can do a multiple classification using the function *'Change functional location with classification' TPCCH MC*.

In communication step *TPCCR CR* (CAD→SAP), the characteristic values assigned in the external system are sent to the SAP System. A characteristic may occur more than once in this list. This is the case if multiple values are assigned to a characteristic.

If no data on characteristic values is sent to the SAP System in this communication step - in other words, if the send string has data length '0' - the functional location is allocated to the class only. No characteristic values are assigned in this case.

In the customizing functions for the CAD interface, you can define whether document numbers can be assigned externally only, internally only, or both.

This function is called from the external application program as follows:

hd = SapTpccr(scode, sndstr, rcvstr, msg)

TPCCH - Changing a Functional Location

The following functions for changing functional locations are supported:

- Changing a Functional Location Without Classification [Seite 261]
- Changing a Functional Location With Classification [Seite 262]

Changing a Functional Location Without Classification

This function is identified by sub-process code *MR*.

It allows you to change data of the functional location, with the exception of classification data. The communication flow of function *TPCCH MR* is described in the table below.

Communication flow for function TPCCH MR

SP code	Direction	Data transferred	Length
MR	→	Key data of the functional location as defined in table TCIU	variable
MD	←	Functional location data in the SAP System as defined in table TCID	variable
MD	→	Changed data of the functional location as defined in table TCIU	variable
MR	←	Function message	variable

When you call the function, the number of the functional location is sent to the SAP System. The SAP interface program then sends the data of the functional location to the external application program as defined in settings in customizing. At the same time, a lock entry is created for the functional location to be changed, in order to prevent concurrent access by other users.

The changes to the functional location data made in the external system are returned to the SAP System in a further communication step. The SAP interface program then triggers the changes in the SAP System.

Changing a Functional Location With Classification

This function is identified by sub-process code *MC*.

The function allows you to change data of the functional location as in function *TPCCH MR*. You can also maintain classification data. You have the following options:

- changing an existing allocation and/or existing characteristic values
- creating an allocation to a new class
- deleting the allocation to a class

You can only create, change, or delete the allocation to one class in one function call. You cannot process several class allocations in one function.

Please note that classification is connected to engineering change management. This means that you can make all changes to classification data with history using either a date or a change number.

The communication flow for function *TPCCH MC* is the same as for function *TPCCH MR* up to communication step *TPCCH MD* (SAP→CAD). When the changed equipment master data is sent to the SAP System in the following communication step *TPCCH MD* (CAD→SAP), you can also send values for the *class type* and *class* whose allocation you want to change.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes to which the functional location is allocated is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class. If a standard class is defined that the allocation is to be changed for, or if the functional location is only allocated to one class, the communication step for transferring a list of classes is no longer necessary. In this case, the information on characteristics and assigned values is sent directly to the external program.

The communication flow of function *TPCCH MC* when changing an existing allocation is described in the table below.

Communication flow for function TPCCH MC for changing an existing allocation

SP code	Direction	Data transferred	Length
MC	→	Key data of the functional location as defined in table TCIU	variable
MD	←	Functional location data in the SAP System as defined in table TCID	variable
MD	→	Changed data of the functional location as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], in which the functional location is classified	variable

RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

After communication step *TPCCH CR* (SAP→CAD), you can either assign values to the characteristics or continue with a different combination of class type, class, and scope.

Communication step *TPCCH RC* (SAP→CAD) only applies if the restrictions sent in communication step *TPCCH MD* (CAD→SAP) produced a list of several classes in which the functional location is classified.

If your change to classification data is an allocation to an additional class, the communication flow is as follows:

Communication flow for function TPCCH MC for creating an allocation to an additional class

SP code	Direction	Data transferred	Length
MC	→	Key data of the functional location as defined in table TCIU	variable
MD	←	Functional location data in the SAP System as defined in table TCID	variable
MD	→	Changed data of the functional location as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], in which the functional location is classified	variable
NC	→	New class type &) New class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable

Changing a Functional Location With Classification

RF	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
CR	→	Changed Assigned Characteristic Values [Seite 120]	variable
MC	←	Function message	variable

If you want to delete an existing allocation to a class, the communication flow is as follows:

Communication flow for function TPCCH MC for deleting an existing allocation to a class

SP code	Direction	Data transferred	Length
MC	→	Key data of the functional location as defined in table TCIU	variable
MD	←	Functional location data in the SAP System as defined in table TCID	variable
MD	→	Changed data of the functional location as defined in table TCIU including class type, class, and scope	variable
RC	←	List of All Classes [Seite 126], in which the functional location is classified	variable
DC	→	Class type to be deleted &) Class to be deleted &)	variable variable
MC	←	Function message	variable

This function is called from the external application program as follows:

hd = SapTpcch(scode, sndstr, rcvstr, msg).

TPCRQ - Displaying a Functional Location

The following functions for displaying functional locations are supported:

- Displaying a Functional Location Without Classification [Seite 266]
- Displaying a Functional Location Without Classification [Seite 266]
- Finding a Functional Location via Matchcode [Seite 269]
- Displaying a Functional Location Hierarchy [Seite 272]
- Displaying a Functional Location with Installed Equipments [Seite 273]

Displaying a Functional Location Without Classification

This function is identified by sub-process code *MR*.

The function allows you to display data of the functional location, with the exception of classification data. The communication flow of function *TPCRQ MR* is described in the table below.

Communication flow for function TPCRQ MR

SP code	Direction	Data transferred	Length
MR	→	Key data of the functional location as defined in table TCIU	variable
MR	←	Functional location data in the SAP System as defined in table TCID	variable

When you call the function, the number of the functional location is sent to the SAP System. The SAP interface program then sends the existing functional location data as defined in settings in customizing to the external application program.

Displaying a Functional Location With Classification

This function is identified by sub-process code *MC*.

The function allows you to display data of the functional location as in function *TPCRQ MR*. You can also display classification data. The communication flow of function *TPCRQ MC* is described in the table below.

Communication flow for function TPCRQ MC

SP code	Direction	Data transferred	Length
MC	→	Key data of the functional location as defined in table TCIU including class type, class, and scope	variable
MC	←	Functional location data in the SAP System as defined in table TCID	variable
RC	←	List of All Classes [Seite 126], in which the functional location is classified	variable
RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable
RC	→	Class type &) Class &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: standard class &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Available Values for Characteristics [Seite 119]	variable

When you call function *TPCRQ MC*, you can send values for the *class type* and *class* whose allocation you want to display.

This data has a restrictive function.

- If you do not enter these values, a data string including all classes to which the functional location is allocated is sent to the external application program.
- If you enter a value for the class type, only the classes of this class type are sent to the external application program.
- There is also a further restriction you can make regarding the standard class.

  In the following situations, the information on characteristics and assigned values is sent directly to the external program:

- You enter the class that the allocation is to be displayed for.
- The restrictions in force when you call the function apply to only one class
- The functional location is only classified in one class.

- You can also enter a date or a change number. Since classification is made with history, the status on a specific date is displayed. If you do not enter a date, the current date is taken as a default.
  - If the restrictions on classification that you send with the function call are such that **several** classes are selected, this list of classes is first sent to the external application program in communication step *TPCRQ RC* (SAP→CAD). In the external system, you select the class you are interested in and return the selected class to the SAP System in communication step *TPCRQ RC* (CAD→SAP).

    You then see the characteristic and characteristic values for the class and scope you require. As shown in the table above, communication step *TPCRQ RC* (CAD→SAP) can be repeated as often as you require. In this way, you can display the classification data of the functional location in all the classes in the previous list. You can also vary the scope. This means that you can only terminate function *TPCRQ MC* with function *BREAK*, because the function does not come to a natural end.
  - If only **one** class is selected, or if the functional location is classified in exactly one class, you can only vary the scope in communication step *TPCRQ RC* (CAD→SAP).

  Please note that you can only use function *TPCRQ RC* (CAD→SAP) to display classification data of the classes selected when you called the function. Classification data for classes that were not selected cannot be displayed. This means that This means that, if the functional location is classified in class types ‘001’ and ‘002’, and you restrict the class type to class type ‘001’ when you call function *TPCRQ MC* (CAD→SAP), the classes of class type ‘002’ are not displayed. You cannot use function *TPCRQ RC* (CAD→SAP) and enter class type ‘002’ to display these classes. You need to call function *TPCRQ MC* again.

Finding a Functional Location via Matchcode

This function is identified by sub-process code *RL*.

The function allows you to find a functional location in the SAP System if you do not know the number that identifies it, but do know other master data, such as the description or cost center. You can also use classification system functions to find functional locations. This function is described in the section on the classification system.

The matchcode functionality allows you to define a matchcode ID such that it starts the function for finding objects via classes or a report, instead of the standard matchcode search function. These matchcode IDs cannot be processed using the CAD interface. If you select one of these matchcode IDs, you see an error message telling you that the matchcode ID cannot be used.

When you use a matchcode, either:

- You do not know the field structure of the selected matchcode ID, or
- You know the field structure of the selected matchcode ID.

The communication flow of function *TPCRQ RL* is described in the tables below.

Communication flow for function TPCRQ RL if you do not know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Selected matchcode ID &) Matchcode sub-ID &)	variable variable
RF	←	First matchcode field &) Length of first matchcode field &) . . . Last matchcode field &) Length of last matchcode field &)	variable variable . . . variable variable
RF	→	Matchcode selection string	variable
CL	←	Functional location data in the SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Functional location data in the SAP System as defined in table TCID	variable

Finding a Functional Location via Matchcode

Communication flow for function TPCRQ RL if you know the field structure of the matchcode ID

SP code	Direction	Data transferred	Length
RL	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Matchcode selection string	variable
CL	←	Functional location data in the SAP System as defined in table TCID	variable
CL	→	None	0
RL	←	Functional location data in the SAP System as defined in table TCID	variable

The matchcode sub-ID is still included in the transferred data strings to ensure compatibility with R/2. In data strings sent from the R/3 System to the external application program, the matchcode sub-ID is SPACE. In data strings sent from the external application program to the SAP System, you can either send the additional separator character or not.

- After you call function *TPCRQ RL*, first the matchcode IDs defined in the SAP plant maintenance system for functional locations are sent to the external application program. For further processing, there is a difference depending on whether you know the field structure of the selected matchcode or not.
  - If you do **not** know the field structure, you send the selected matchcode ID to the SAP System in the next communication step in order to get the field structure.
  - However, if you know the field structure, you can use the next communication step to send the matchcode selection string. To do this, you use the matchcode identifier (usually the ‘0’ character). The matchcode identifier is transferred to the external application program using system function *SYSRQ PA*.
- When you use matchcodes to find functional locations, please note that the master record of each functional location found by the matchcode search is read in order to procure the data that is not stored in matchcode records. However, this may lead to runtime problems if the matchcode selection criteria are not specific enough and therefore a large number of functional location records is selected. For this reason, control table TCIS contains parameter TCIS-MCMAX, which defines the maximum number of matchcode records, after which communication with the external application program is triggered. If such a communication is necessary, you use sub-process code *CL*. If you call function *TPCRQ CL,* you can then request the following matchcode records.

As of release 3.0, all matchcode selection functions were changed such that functions *TPCRQ ID* and *TPCRQ RF* can be called in any communication step. This has the advantage that you do not need to start again by calling function *TPCRQ RL* if your matchcode search is unsuccessful. Instead, you can enter new values in the matchcode fields directly if you want to use the current matchcode ID, or switch to another matchcode ID. However, this means that the matchcode

selection functions no longer come to a natural end. As of release 3.0, you terminate these functions using the *BREAK* function

Displaying a Functional Location Hierarchy

This function is identified by sub-process code *RH*.

The function allows you to explode the hierarchy containing the current functional location. You can set up a functional location hierarchy using function *TPCCR MR, TPCCH MC* or *TPCCH MR* by entering a value for the superior functional location. You can restrict explosion of the hierarchy to one direction or other. You can also define the number of levels of the hierarchy you want to explode.

The communication flow of function *TPCRQ RH* is described in the table below.

Communication flow for function TPCRQ RH

SP code	Direction	Data transferred	Length
RH	→	Number of the functional location &) Effectivity date &) Number of explosion levels upward &) Number of explosion levels downward &)	variable variable variable variable
RH	←	Functional location data in the SAP System as defined in table TCID for each functional location found in the hierarchy	variable

When you call the function, you must send the number of the functional location and the effectivity date for which you want to explode the hierarchy. If you do not enter a date, the current date is used as a default. If you do not enter the number of explosion levels downward, the number is set to ‘99’. However, if you do not enter the number of explosion levels upward, the number is set to ‘0’. This means that the hierarchy above the current equipment is not exploded.

Displaying a Functional Location with Installed Equipments

This function is identified by sub-process code *GM*.

The function allows you to explode the hierarchy containing the current functional location and explode the equipment hierarchies of the equipments installed at the functional locations. You can set parameters for this function such that no functional location hierarchy is exploded or no equipment hierarchies are exploded. You can restrict explosion of the functional location hierarchy to one direction or the other. You can also define the number of levels of the hierarchy you want to explode.

The communication flow of function *TPCRQ GM* is described in the table below.

Communication flow for function TPCRQ GM

SP code	Direction	Data transferred	Length
GM	→	Number of the functional location &) Effectivity date &) Number of explosion levels upward &) Number of explosion levels downward &) Indicator: with functional location hierarchy &) Indicator: with equipment hierarchy &) Indicator: with materials/PM assemblies &)	variable variable variable variable variable variable variable
IB	←	Functional location data in the SAP System as defined in table TCID for each functional location found in the hierarchy	variable
IP	←	Equipment data in the SAP System as defined in table TCID for each equipment found in the hierarchy	variable
GM	←	Material data in the SAP System as defined in table TCID for each material found in the hierarchy	variable

When you call the function, you must send the number of the functional location and the effectivity date for which you want to explode the hierarchy. If you do not enter a date, the current date is used as a default. If you do not enter the number of explosion levels downward, the number is set to ‘99’. However, if you do not enter the number of explosion levels upward, the number is set to ‘0’. This means that the hierarchy above the current equipment is not exploded.

This function is called from the external application program as follows:

hd = SapTpcrq(scode, sndstr, rcvstr, msg).

Classification System Functions

The following classification system functions are supported:

SLCLS - Finding Classes - Navigating in a Class Network [Seite 275]

SLOBJ - Finding Objects in Classes [Seite 285]

SLCLS - Finding Classes - Navigating in a Class Network

The CAD interface supports a range of functions for finding classes and for navigating in a class network:

- Finding Classes by Catchwords (Matchcode) [Seite 276]
- Finding Classes by Generic Entry [Seite 279]
- Finding Classes without a Superior Class [Seite 281]
- Displaying Superior and Subordinate Classes for a Class [Seite 283]

Finding Classes by Catchwords (Matchcode)

The function call *SLCLS GM* identifies the function for finding classes by catchwords.

The communication flow is described in the tables below. As in all matchcode selection functions, the procedure depends on whether you know the field structure of the matchcode you are using or not.

Communication flow for function SLCLS GM if you do not know the field structure of the matchcode

SP code	Direction	Data transferred	Length
GM	→	None	0
ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Selected matchcode ID &) Matchcode sub-ID &) (optional)	variable variable
RF	←	First matchcode field &) Length of first matchcode field &) . . . Last matchcode field &) Length of last matchcode field &)	variable variable . . . variable variable
RF	→	Matchcode selection string	variable
GM	←	Class type &) Class &) Catchword &) This sequence is repeated for each selected class.	variable variable variable

Communication flow for function SLCLS GM if you know the field structure of the matchcode

SP code	Direction	Data transferred	Length
GM	→	None	0

ID	←	First matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &) . . . Last matchcode ID &) Matchcode sub-ID &) Description of matchcode ID &)	variable variable variable . . . variable variable variable
ID	→	Matchcode selection string	variable
GM	←	Class type &) Class &) Catchword &) This sequence is repeated for each selected class.	variable variable variable

The matchcode sub-ID is still included in the transferred data strings to ensure compatibility with R/2. In data strings sent from the R/3 System to the external application program, the matchcode sub-ID is SPACE. In data strings sent from the external application program to the SAP System, you can either send the additional separator character or not.

- After you call function *SLCLS GM,* first all the matchcode IDs defined in the SAP classification system are sent to the external application program. For further processing, there is a difference depending on whether you know the field structure of the selected matchcode or not.
  - If you do **not** know the field structure, you send the selected matchcode ID to the SAP System in the next communication step in order to get the field structure.
  - However, if you know the field structure, you can use the next communication step to send the matchcode selection string. To do this, you use the matchcode identifier (usually the ‘0’ character). The matchcode identifier is transferred to the external application program using system function *SYSRQ PA*.

When you find classes via matchcode, there is no sub-function SLCLS CL as for matchcode selection of other objects. This is because no further master data is read for the classes found via matchcode. This means that there is no problem with runtimes, so you do not need to interrupt the function after a certain number of selected objects.

As of release 3.0, all matchcode selection functions have been changed such that functions *SLCLS ID* and *SLCLS RF* can be called in any communication step. This has the advantage that you do not need to start again by calling function *SLCLS GM* if your matchcode search is unsuccessful. Instead, you can enter new values in the matchcode fields directly if you want to use the current matchcode ID, or switch to another matchcode ID. However, this means that the matchcode selection functions no longer come to a natural end. As of release 3.0, you terminate these functions using the *BREAK* function

Communication example:

You want to find all classes with the catchword ‘Motor’. To illustrate the change described above, first we assume that you enter the search string ‘MOTOR’ in upper case letters. However, since matchcodes are case sensitive, no class is found. Instead of starting again with function *SLCLS GM*, you can now continue immediately with function *SLCLS RF* and the new search string ‘Motor’.

SLCLSGM 000000

Finding Classes by Catchwords (Matchcode)

SLCLSID00000000154 A@@Über Schlagwort suchen@B@@Über Klassenn ummer suchen@C@@Über Klassengruppe suchen@D@@Über DIN-Klasse suc hen@E@@Über Norm suchen@F@@Über Dokument suchen@

SLCLSID 000002 A@

SLCLSRF00000000057 Schlagwort@0040@Sprache@0001@Klassenart@0003

@Klasse@0018@

SLCLSRF 000005 MOTOR

SLCLSGM04046000000 ECH013 Es wurden keine Matchcodesätze gefundenSLCLSRF 000005 Motor

SLCLSGM00000000553 002@MOTOREN@Motor@300@MOT_K@Motor@200@BD_MOTOR@Motor für den Lux usschlitten@...

BREAK 000000

BREAK 00087000000 SCH003 Die Funktion ‘Suchen Klasse über Schla gwort (Matchcodesuche) ‘ wurde abgebrochen

Im nächsten Kommunikationsbeispiel soll alternativ der Verlauf der Matchcodesuche gezeigt werden, der sich ergibt, wenn der Feldaufbau der verwendeten Matchcode-ID bekannt ist:

SLCLSGM 000000

SLCLSID00000000154 A@@Über Schlagwort suchen@B@@Über Klassenn ummer suchen@C@@Über Klassengruppe suchen@D@@Über DIN-Klasse suc hen@E@@Über Norm suchen@F@@Über Dokument suchen@

SLCLSID 000008 =A.MOTOR

SLCLSGM04046000000 ECH013 Es wurden keine Matchcodesätze gefunden

SLCLSID 000008 =A.Motor

SLCLSGM00000000553 002@MOTOREN@Motor@300@MOT_K@Motor@200@BD_MOTOR@ Motor für den Luxusschlitten@...

BREAK 000000

BREAK 00087000000 SCH003 Die Funktion ‘Suchen Klasse über Schla gwort (Matchcodesuche) ‘ wurde abgebrochen

Finding Classes by Generic Entry

Function call *SLCLS HM* identifies the function for finding classes by generic entry.

The communication flow is described in the table below.

Communication flow for function SLCLS HM

SP code	Direction	Data transferred	Length
HM	→	Class type &)	variable
		Class &)	variable
		Effectivity date &)	variable
		Language &)	variable
		Indicator: with allocations &)	variable
HM	←	Class type &)	variable
		Class &)	variable
		Catchword &)	variable
		Class status &)	variable
		Valid from date &)	variable
		Valid to date &)	variable
		Indicator: same classification &)	variable
		Number of allocations &)	variable
		Indicator: DIN class &)	variable
		Name of standard &)	variable
		Standard number &)	variable
		Standard description &)	variable
		Standard title 1 &)	variable
		Standard title 2 &)	variable
		Issue date &)	variable
		Version date &)	variable
		Version number &)	variable
		Characteristics table &)	variable
		Document type &)	variable
		Document number &)	variable
		Document part &)	variable
		Document version &)	variable
		This sequence is repeated for each selected class.	

You can send both the class type and the class either exactly or generically in the function call. '+' represents any single character and '*' represents any character string. If you do not send the effectivity date or the language, the current date and the logon language are used. If you set the 'With allocations' indicator, only classes that have objects allocated to them are found. If you do not set this indicator, classes with no objects allocated to them are also found.

Communication example:

You want to find all classes of class type MWV whose name starts with PP- and that have objects allocated to them. The selection date is 01.09.1995.

SLCLSHM 000023 MWV@PP-*@01.09.1995@@X@

SLCLSHM00000000406 MWV@PP-LOESBAREVERB@Lösbare Verbindungen@1@28.08.1995@31.12.9999@X@00001@@@@@@@@00000000@00@@

Finding Classes by Generic Entry

@@@@MWV@PP-MUTTERN@Muttern metrisch@1@28.
08.1995@31.12.9999@X@00001@@@@@@@@00000000@00@@@@@@MWV@PP-
SCHRAUBEN@Schrauben
metrisch@1@28.08.1995@31.12.9999@X@00001@@@@@@@@00000000@00@@DRW@
PP-VERSCHIEDENE_SCHRAUBEN@000@00@MWV@PP-
VERBELEMENTE@Verbindungselemente@1@28.08.1995@31.12.9999@E@00001@@@@@
@@@00000000@00@@@@@@

Finding Classes without a Superior Class

This function is one of two functions in the CAD interface that allows you to analyze the structure of the class network. Function call *SLCLS HS* identifies this function.

The function allows you to find classes that do not have superior classes. These classes are effectively the top end of the class network. The communication flow is described in the table below.

Communication flow for function SLCLS HS

SP code	Direction	Data transferred	Length
HS	→	Class type &)	variable
		Class &)	variable
		Effectivity date &)	variable
		Language &)	variable
HS	←	Class type &)	variable
		Class &)	variable
		Catchword &)	variable
		Class status &)	variable
		Valid from date &)	variable
		Valid to date &)	variable
		Indicator: same classification &)	variable
		Number of allocations &)	variable
		Indicator: DIN class &)	variable
		Name of standard &)	variable
		Standard number &)	variable
		Standard description &)	variable
		Standard title 1 &)	variable
		Standard title 2 &)	variable
		Issue date &)	variable
		Version date &)	variable
		Version number &)	variable
		Characteristics table &)	variable
		Document type &)	variable
		Document number &)	variable
		Document part &)	variable
		Document version &)	variable
		This sequence is repeated for each selected class.	

You can send both the class type and the class either exactly or generically in the function call. '+' represents any single character and '*' represents any character string. If you do not send the effectivity date or the language, the current date and the logon language are used.

Communication example:

You want to find all classes of class type MWV whose name starts with PP- and that have no superior class. The selection date is 01.09.1995.

SLCLSHS 000021 MWV@PP-*@01.09.1995@@

Finding Classes without a Superior Class

SLCLSHS00000000096 MWV@PP-VERBELEMENTE@Verbindungselemente@1@28.08.1995@31.12.9999@E@00001@@@@@@@@@00000000@00@@@@@@

Displaying Superior and Subordinate Classes for a Class

This function is one of two functions in the CAD interface that allows you to analyze the structure of the class network. Function call *SLCLS RH* identifies this function.

The function allows you to explode the class network structure above and below the current class. You can restrict explosion of the hierarchy to one direction or other. You can also define the number of levels of the hierarchy you want to explode. The communication flow is described in the table below.

Communication flow for function SLCLS RH

SP code	Direction	Data transferred	Length
RH	→	Class type &) Class &) Effectivity date &) Language &) Number of explosion levels upward &) Number of explosion levels downward &)	variable variable variable variable variable variable
RH	←	Level &) Indicator: Subordinate class &) Superior class &) Subordinate class &) Catchword of superior class&) Catchword of subordinate class &) This sequence is repeated for each selected class.	variable variable variable variable variable variable

When you call the function, you must send the class type and class for which you want to explode the structure. If you do not send the validity date or the language, the current date and the logon language are used.

- If you do not enter the number of explosion levels downward, the number is set to ‘1’.
- However, if you do not enter the number of explosion levels upward, this number is set to ‘0’. This means that the structure above the current class is not exploded.

Communication example:

In the above example, class PP-VERBELEMENTE was identified as a class without a superior class. This means that there are no classes above this class in the class network. In this example, you want to determine all the classes directly below class PP-VERBELEMENTE. For this reason, the value for explosion levels downward is ‘1’. The selection date is 01.09.1995.

SLCLSRH 000036 MWV@PP-VERBELEMENTE@01.09.1995@@

0@1@

SLCLSRH00000000234 0@X@PP-VERBELEMENTE@PP-VERBELEMENTE@Verbindungselemente@Verbindungselemente@1@X@PP-VERBELEMENTE@PP-LOESBAREVERB@Verbindungselemente@Lösbare Verbindungen@1@X@PP-VERBELEMENTE@PP-UNLOESBAREVERB@Verbindungselemente@Unlösbare Verbindungen@

This function is called from the external application program as follows:

Displaying Superior and Subordinate Classes for a Class

hd = SapSlcls (scode, sndstr, rcvstr, msg).

SLOBJ - Finding Objects in Classes

The CAD interface supports a number of functions for finding objects in classes:

- Displaying the Characteristics of a Class [Seite 286]
- Displaying All Objects in a Class [Seite 287]
- Finding Objects via Classes [Seite 290]
- Setting a New View in Table INDX [Seite 294]

Displaying the Characteristics of a Class

This function is used to display the characteristics of a class. It is identified by function call *SLOBJ RF*.

The communication flow is described in the table below.

Communication flow for function SLOBJ RF

SP code	Direction	Data transferred	Length
RF	→	Class type &) Class &) Effectivity date &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable

When you call the function, you can send values for the scope. The scope is used to select the relevant characteristics of the class. You can, for example, display only the required characteristics, or only the optional characteristics. You can also hide or show the characteristics that are inherited from another class.

Communication example:

You want to display all the characteristics of class PP-SKTSCHRAUBE, including inherited characteristics.

SLOBJRF 000038 MWV@PP-SKTSCHRAUBE@01.09.1995@X@X@@@X@

SLOBJRF00000000810 MWV@PP-SKTSCHRAUBE@Sechskantschrauben DIN_931@@@PP-GEWINDEDURCHMESSER@Gewindedurchmesser@@@@NUM@X@X@@ 5@2@@0@0@___,__@X@mm@@@@@@@@@@@@@@@@@PP-SCHRAUBENLAENGE@Schraubenlänge@@@@NUM@X @X@@5@ 2@@0@0@___,__@X@mm@@@@@@@@@@@@@@@@...

Displaying All Objects in a Class

This function allows you to display all the objects classified in a class. The function call *SLOBJ RL* is used to identify this function.

The function is effectively a find objects function without selection criteria, so that all objects are always displayed. Unlike *SLOBJ RI,* function *SLOBJ RL* does not differentiate according to classification status. All objects allocated to the class are displayed, regardless of their status. The communication flow is described in the table below.

Communication flow for function SLOBJ RL

SP code	Direction	Data transferred	Length
RL	→	Class type &)	variable
		Class &)	variable
		Effectivity date &)	variable
		Indicator: required characteristics &)	variable
		Indicator: optional characteristics &)	variable
		Indicator: characteristics with assigned values &)	variable
		Indicator: characteristics without assigned values &)	variable
		Indicator: inherited characteristics &)	variable
RF	←	Characteristics Data for a Class [Seite 116]	variable

Displaying All Objects in a Class

RL	←	Object ID (‘K’: class, ‘O’: object) &)	variable
		Object table &)	variable
		Object key field 01 &)	variable
		Object key field 02 &)	variable
		Object key field 03 &)	variable
		Object key field 04 &)	variable
		Object key field 05 &)	variable
		Object key field 06 &)	variable
		Object key field 07 &)	variable
		Object key field 08 &)	variable
		Object key field 09 &)	variable
		Object key field 10 &)	variable
		Object description &)	variable
		Classification status &)	variable
		Name of first characteristic &)	variable
		Value for first characteristic &)	variable
		.	.
		.	.
		.	.
		Name of last characteristic &)	variable
		Value for last characteristic &) &)	variable
		This sequence is repeated for each selected object.	

When you display values assigned to characteristics, please note that a characteristic name can occur more than once for a selected object. This is always the case if multiple values are assigned to the characteristic.

The field *Object table* is used to identify the type of the selected object.

Object tables

Object table	Description
AENR	Change master record
DRAW	Document info record
EQUI	Equipment
IFLOT	Functional location
MARA	Material master record

The object table is only displayed for *real* objects, that is to say, only if the object ID has the value '**O**'.

If the object ID of the selected object has the value '**K**', the object is a class. In this case, no value is transferred for the object table.

After the last characteristic value for an object, an additional separator character is sent. This separates the characteristics belonging to different objects.

Communication example:

You want to display all objects classified in class PP-SKTSCHRAUBE of class type MWV. The selection date is 01.09.1995. You want to display all the characteristics - including inherited characteristics. You want to display all characteristics, including inherited characteristics.

```
SLOBJRL     000040 MWV@PP-SKTSCHRAUBE@01.09.1995@X@

X@X@X@X@

SLOBJRF00000000810XMWV@PP-SKTSCHRAUBE@SechskantschraubenDIN_931@@@PP-
GEWINDEDURCHMESSER@Gewindedurchmesser@@@@NUM@X@X@@5@2@@0@0@___,__@X@mm@
@@@@@@@@@@@@@@@@PP-
SCHRAUBENLAENGE@Schraubenlänge@@@@NUM@X@X@@5@2@@0@0@___,__@X@mm@@@@@@@@
@@@@@@@@...

SLOBJRL00000000774 O@MARA@50004194@@@@@@@@@@Sechskantschraube@PP-
LOESBAREVERB@@PP-GEWINDEART@Trapezgewinde       DIN 103@PP-NORM@@PP-
GEWINDEDURCHMESSER@@PP-FESTIGKEITSKLASSE@5,6@PP-GEWINDEBEZEICHNUNG@@PP-
SCHLUESSELWEITE@@PP-SCHRAUBENKOPF@@PP-SCHRAUBENLAENGE@@PP-
KOPFHOEHE@@@O@MARA@
```

Finding Objects via Classes

This function allows you to select objects that are classified in a class. The function is identified by function call *SLOBJ RI*.

The communication flow is described in the table below.

Communication flow for function SLOBJ RI

SP code	Direction	Data transferred	Length
RI	→	Class type &) Class &) Effectivity date &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: objects with unassigned characteristics &) Indicator: objects with locked allocations &) Indicator: objects with incomplete allocations &)	variable variable variable variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
RC	→	Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &) Indicator: objects with unassigned characteristics &) Indicator: objects with locked allocations &) Indicator: objects with incomplete allocations &)	variable variable variable variable variable variable variable variable
RF	←	Characteristics Data for a Class [Seite 116]	variable
CR	←	Values for Characteristics [Seite 119] (suggestion)	variable
CR	→	Name of characteristic &) Selection criterion for characteristic &) Comparison type for characteristic &) Name of an additional characteristic &) Selection criterion for characteristic &) Comparison type for characteristic &) . . .	variable variable variable variable variable variable . . .

RI	←	Object ID (‘K’: class, ‘O’: object) &)	variable
		Object table &)	variable
		Object key field 01 &)	variable
		Object key field 02 &)	variable
		Object key field 03 &)	variable
		Object key field 04 &)	variable
		Object key field 05 &)	variable
		Object key field 06 &)	variable
		Object key field 07 &)	variable
		Object key field 08 &)	variable
		Object key field 09 &)	variable
		Object key field 10 &)	variable
		Object description &)	variable
		Classification status &)	variable
		Name of first characteristic &)	variable
		Value for first characteristic &)	variable
		.	.
		.	.
		.	.
		Name of last characteristic &)	variable
		Value for last characteristic &) &)	variable
		This sequence is repeated for each selected object.	

The communication flow of this function is far more flexible than the table above would suggest.

- First, the values for the class type and class name are sent to the SAP System with function call *SLOBJ RI*.

  You can also define the selection date. If you do not send a selection date, the current date is taken as a default.

  You can also define the scope to be used in the function call. For example, the scope determines whether inherited characteristics are shown or hidden.

  There are three more indicators for determining which classification status an object must have in order to be selected. You can define whether objects with no values assigned to the characteristics, objects with locked allocation, or objects with incomplete allocations are selected.

- In the next communication step, two data strings are sent to the external application with function codes *SLOBJ RF* and *SLOBJ CR*. These data strings contain technical information on the relevant characteristics and the default values for the characteristics.

- Then you can continue from the external application program with two different function calls.

  – If you want to change the values for the scope or the indicators described above again, you can do this with function call *SLOBJ RC*. The SAP System sends the characteristics as defined in the changed scope and their default values. You can repeat this as often as you like.

  – If you have made the settings you require, you can send the selection criteria to the SAP System with function call *SLOBJ CR*. You can, of course, display the allowed values for one or more characteristics using functions *HLPRQ SE* and *HLPRQ SC* in between times.

Finding Objects via Classes

- The selection criteria are assumed to be in the form used in the SAP System. The comparison type is required for each selection criteria. You can find values inclusively with **INCL**, exclusively with **EXCL**, inclusive of a string with **SCAN**, and exclusive of a string with **EXSC**.
- If you do not make an entry, **INCL** is assumed as the default value.

  The values have the following meaning:

  INCL

  Only objects whose characteristic value matches the selection criteria

  EXCL

  Only objects whose characteristic value does not match the selection criteria

  SCAN

  Only objects whose characteristic value contains the selection criteria

  EXSC

  Only objects whose characteristic value does not contain the selection criteria

- In the following communication step *SLOBJ RI,* the SAP-System sends the classes or objects that match the selection criteria to the external application program.

  You can continue with communication steps *SLOBJ CR* or *SLOBJ RC* in order to define new selection criteria or even a new scope.

- After the last characteristic value for an object, an additional separator character is sent. This separates the characteristics belonging to different objects.

Since this find object function does not come to a natural end, you need to use the function *BREAK* to exit this function as required.

Communication example:

You want to display the flat nuts classified in class PP-MUTTERN of class type MWV that have a diameter of 5 mm - 7 mm. The selection date is 01.09.1995. You want to display all the characteristics - including inherited characteristics. You want to display all characteristics, including inherited characteristics. You do not want to display characteristics without assigned values or locked or incomplete allocations.

SLOBJRI 000040 MWV@PP-MUTTERN@01.09.1995@X@X@X@X@X@@@@

SLOBJRF00000000243XMWV@PP-MUTTERN@Muttern metrisch@@@PP-NORM@Norm@@@@CHAR@X@@@10@0@@0@0@@X@@@@@@@@@@@@@@@@@@PP-MUTTERFORM@Mutterform@@@@CHAR@X@@@20@0@@0@0@@X@@@@@@@@@@@@@@@@@PP-GEWINDEDURCHMESSER@Gewindedurchmesser@@@@NUM@X@X@@5@2@@0@0@_ __,__@X@mm@@@@@@@@@@@@@@@@

SLOBJCR00000000024 PP-MUTTERFORM@Sechskant@

SLOBJCR00000000064 PP-MUTTERFORM@Flach@INCL@PP-GEWINDEDURCHMESSER@5 mm - 7 mm@INCL@

SLOBJRI00000000196 O@MARA@R-12@@@@@@@@@@Mutter@1@PP-GEWINDEDURCHMESSER@5,00@PP-MUTTERFORM@Flach@PP-NORM@Flachmuttern@@O@MARA@J-11@@@@@@@@@@Mutter@1@PP-GEWINDEDURCHMESSER@7,00@PP-MUTTERFORM@Flach@PP-NORM@Flachmuttern@

BREAK 00000000000

BREAK 07000000000 SCH003 Die Funktion ‘Selektion Objekte überKlasse ‘ wurde abgebrochen

Setting a New View in Table INDX

You can use this function to store the scope used in classification system functions in the SAP System long term. Function call *SLOBJ SC* is used to identify this function.

The scope sent to the SAP System in this function is used as a default setting. The scope is specific to one class type and one user name. The communication flow is described in the table below.

Communication flow for function SLOBJ SC

SP code	Direction	Data transferred	Length
SC	→	Class type &) Indicator: required characteristics &) Indicator: optional characteristics &) Indicator: characteristics with assigned values &) Indicator: characteristics without assigned values &) Indicator: inherited characteristics &)	variable variable variable variable variable variable
SC	←	Message	variable

Communication example:

You want to set the default scope for class type MWV for the user that is currently logged on.

SLOBJSC 000012 MWV@X@@X@@X@

SLOBJSC00032000000 SCH089 Der Scope wurde gesichert

This function is called from the external application program as follows:

hd = SapSlobj (scode, sndstr, rcvstr, msg).

Initializing Class Data

The CAD interface does not display out-of-date data when the classification data is accessed repeatedly as long as the function SLOBJ IN was carried out first.

This function resets the buffer when the classification data is read again from the data base.

The function SLOBJ IN does not call any data, rather it intitalizes them.

Table TCIQ

Verify whether the following entries are in the table TCIQ:

Process code	Sub-process code	Direction	Counter
SLOBJ	IN	D	01
SLOBJ	IN	U	01

The following fields remain empty:

- FIELDSET
- Screen number
- TABLE"

Customizing for CAD Integration

Enter the process code SLOBJ IN in Customizing; under *Cross-application components* → *CAD Integration* → *General Data* → *Determine functions.*

Office Functions

The CAD interface supports a number of functions for processing SAPoffice objects. For more information, see:

SMAIL - Office Functions [Seite 297]

SMAIL - Office Functions

The CAD interface supports a number of Office functions. These functions allow user-to-user communication between the external application program and the SAP System.

The following functions are supported in release 3.0:

- Reading Your Inbox [Seite 298]
- Reading Your Outbox [Seite 299]
- Reading Your Mail Trash [Seite 300]
- Reading a Document [Seite 301]
- Reading the Attachments to a Document [Seite 303]
- Setting a Document to “In Process” Status [Seite 304]
- Setting a Document to “Done” Status [Seite 305]
- Displaying the Recipient List for a Document [Seite 307]
- Sending a Document [Seite 308]
- Finding an SAP User Name Generically [Seite 309]

Reading Your Inbox

This function is identified by sub-process code *IB*.

The function allows you to read the contents of the Office inbox belonging to the user you used to logon to the SAP System via the CAD interface.

The communication flow of function *SMAIL IB* is described in the table below.

Communication flow for function SMAIL IB

SP code	Direction	Data transferred	Length
IB	→	None	0
IB	←	Data for each document in Office inbox as defined in table TCID	variable

After you call the function, the SAP interface program sends a list of all documents in the Office inbox. You can define which data fields are transferred for each document using the customizing functions for the CAD interface. Data field *SOFOLENTI1-DOC_ID* must always be transferred. This field contains the key of a document, which you always need to use when you want to process a document.

Reading Your Outbox

This function is identified by sub-process code *OB*.

The function allows you to read the contents of the Office outbox belonging to the user you used to logon to the SAP System via the CAD interface.

The communication flow of function *SMAIL OB* is described in the table below.

Communication flow for function SMAIL OB

SP code	Direction	Data transferred	Length
OB	→	None	0
OB	←	Data for each document in Office outbox as defined in table TCID	variable

After you call the function, the SAP interface program sends a list of all documents in the Office outbox. You can define which data fields are transferred for each document using the customizing functions for the CAD interface. Data field *SOFOLENTI1-DOC_ID* must always be transferred. This field contains the key of a document, which you always need to use when you want to process a document.

Reading Your Mail Trash

This function is identified by sub-process code *WB*.

The function allows you to read the contents of the Office trash belonging to the user you used to logon to the SAP System via the CAD interface.

The communication flow of function *SMAIL WB* is described in the table below.

Communication flow for function SMAIL WB

SP code	Direction	Data transferred	Length
WB	→	None	0
WB	←	Data for each document in Office trash as defined in table TCID	variable

After you call the function, the SAP interface program sends a list of all documents in the Office trash. You can define which data fields are transferred for each document using the customizing functions for the CAD interface. Data field *SOFOLENTI1-DOC_ID* must always be transferred. This field contains the key of a document, which you always need to use when you want to process a document.

Reading a Document

This function is identified by sub-process code *RM*.

The function allows you to read a document in the Office inbox, Office outbox, or Office trash. However, first you must call the function for reading the relevant folder: *SMAIL IB*, *SMAIL OB,* or *SMAIL WB*. These functions serve to send the keys of the individual documents to the external application program. These keys are required in function *SMAIL RM* in order to identify the document you want to process.

The communication flow of function *SMAIL RM* is described in the table below.

Communication flow for function SMAIL RM

SP code	Direction	Data transferred	Length
RM	→	DOCUMENT_ID	variable
MH	←←	Document header data as defined in table TCID	variable
MT	←	First text line of document &) . . . Last text line of document &)	variable . . . variable
MA	←	ID of first attachment to document &) Type of attachment &) Name of document of attachment &) Title of attachment &) . . . ID of last attachment to document &) Type of attachment &) Name of document of attachment &) Title of attachment &)	variable variable variable variable . . . variable variable variable variable

Reading a Document

MP	←	Name of first parameter &)	variable
		Option of parameter &)	variable
		Lowest value of parameter &)	variable
		Highest value of parameter &)	variable
		.	.
		.	.
		.	.
		Name of last parameter &)	variable
		Option of parameter &)	variable
		Lowest value of parameter &)	variable
		Highest value of parameter &)	variable
MR	←	Name of first parameter &)	variable
		Value of parameter &)	variable
		.	.
		.	.
		.	.
		Name of last parameter &)	variable
		Value of parameter &)	variable
RF	→	None	0
RF	←	Function message	variable

When you call the function, the DOCUMENT_ID of the document you want to read is sent to the SAP System. The data of the document is then sent to the external application program in a sequence of five data strings.

- The first data string contains general information on the document, such as the name of the sender or the date on which the document was created.
- The second data string transfers the actual text information of the document. The individual text lines are separated by separator characters.
- The third data string contains the list of attachments to the document.
- The following two data strings contain the processing parameters linked to the document. This allows you to transfer processing parameters and actions as well as text information.

When the document has been received and read in the external system, a confirmation can be sent to the SAP System using function call *SMAIL RF*. The document is then marked as viewed in SAPoffice. This is necessary because only documents that are marked as *viewed* can be deleted. The sender of the document can also see that the document has been read.

Reading the Attachments to a Document

This function is identified by sub-process code *MB*.

It allows you to read the attachments to a document in the Office inbox, Office outbox, or Office trash. However, first you must call the function for reading the relevant folder: *SMAIL IB*, *SMAIL OB,* or *SMAIL WB*. These functions serve to send the keys of the individual documents to the external application program. You need not read the document to which the attachment belongs beforehand. However, when you read a document, the keys of the attachments belonging to the document are transferred to the external application program.

The communication flow of function *SMAIL MB* is described in the table below.

Communication flow for function SMAIL MB

SP code	Direction	Data transferred	Length
MB	→	ATTACH_ID	variable
MC	←	Header data of attachment as defined in table TCID	variable
MB	←	First text line of attachment &) . . . Last text line of attachment &)	variable . . . variable

When you call the function, the ATTACH_ID of the attachment you want to read is sent to the SAP System. The data of the attachment is then sent to the external application program in a sequence of two data strings.

- The first data string contains general information on the attachment, such as the title or the name of the user who created the attachment.
- The second data string transfers the actual text information of the attachment. The individual text lines are separated by separator characters.

Setting a Document to “In Process” Status

This function is identified by sub-process code *MP*.

The function allows you to set the status of a document in the Office inbox to ‘in process’. However, first you have to read the Office inbox using function *SMAIL IB*. These functions serve to send the keys of the individual documents to the external application program.

The communication flow of function *SMAIL MP* is described in the table below.

Communication flow for function SMAIL MP

SP code	Direction	Data transferred	Length
MP	→	DOCUMENT_ID	variable
MP	←	Function message	variable

When you call the function, the DOCUMENT_ID of the document whose status you want to set to ‘in process’ is sent to the SAP System.

Setting a Document to “Done” Status

This function is identified by sub-process code *DO*.

It allows you to set the status of a document in the Office inbox to ‘done’. However, first you have to read the Office inbox using function *SMAIL IB*. These functions serve to send the keys of the individual documents to the external application program.

The communication flow of function *SMAIL DO* is described in the table below.

Communication flow for function SMAIL DO

SP code	Direction	Data transferred	Length
DO	→	DOCUMENT_ID	variable
DO	←	Function message	variable

When you call the function, the DOCUMENT_ID of the document whose status you want to set to ‘done’ is sent to the SAP System.

Deleting a Document

This function is identified by sub-process code *DL*.

The function allows you to delete a document in the Office inbox or Office outbox. However, first you have to read the Office inbox or the Office outbox using function *SMAIL IB* or *SMAIL OB* respectively. These functions serve to send the keys of the individual documents to the external application program. The document you delete with function *SMAIL DL* is moved to the Office trash.

The communication flow of function *SMAIL DL* is described in the table below.

Communication flow for function SMAIL DL

SP code	Direction	Data transferred	Length
DL	→	DOCUMENT_ID	variable
DL	←	Data for each document in Office inbox or Office outbox as defined in table TCID	variable

When you call the function, the DOCUMENT_ID of the document you want to delete is sent to the SAP System. The document is deleted from the folder and placed in the Office trash. A message is then sent to the external application program with a list of the documents remaining in the folder.

Displaying the Recipient List for a Document

This function is identified by sub-process code *RL*.

The function allows you to display the recipient list for a document in the Office inbox, Office outbox, or Office trash. However, first you must call the function for reading the relevant folder: *SMAIL IB*, *SMAIL OB,* or *SMAIL WB*. These functions serve to send the keys of the individual documents to the external application program.

The communication flow of function *SMAIL RL* is described in the table below.

Communication flow for function SMAIL RL

SP code	Direction	Data transferred	Length
RL	→	DOCUMENT_ID	variable
RL	←	Data for each recipient of the document as defined in table TCID	variable

When you call the function, the DOCUMENT_ID of the document whose recipient list you want to display is sent to the SAP System. The list of recipients is sent to the external application program.

Sending a Document

This function is identified by sub-process code *SM*.

The function allows you to send a document to SAPoffice users in the SAP System.

The communication flow of function *SMAIL SM* is described in the table below.

Communication flow for function SMAIL SM

SP code	Direction	Data transferred	Length
SM	→	Document header data as defined in table TCIU	variable
TD	→	First text line &) . . . Last text line &)	variable . . . variable
RC	→	Data for each recipient of the document as defined in table TCIU	variable
SM	←	Function message	variable

When you call the function, the document header data is sent to the SAP System. The second data string contains the actual text information. All text lines must be separated by a separator character. The third data string contains the data for the recipient to whom you want to send the document. For each recipient, you can set a range of flags, such as ‘Express’, ‘Copy’, ‘Reply required’, and so on.

Finding an SAP User Name Generically

This function is identified by sub-process code *UL*.

The function allows you to find SAP user names generically using the address management functionality.

The communication flow of function *SMAIL UL* is described in the table below.

Communication flow for function SMAIL UL

SP code	Direction	Data transferred	Length
UL	→	Generic user name	variable
UL	←	User ID of first Office user &)	variable
		SAP name of Office user &)	variable
		Name of Office user &)	variable
		Name in address management &)	variable
		Indicator: user deleted &)	variable
		.	.
		.	.
		.	.
		User ID of last Office user &)	variable
		SAP name of Office user &)	variable
		Name of Office user &)	variable
		Name in address management &)	variable
		Indicator: user deleted &)	variable

When you call the function, a generic search string for the user name is sent to the SAP System. A list of selected SAP user names is sent to the external application program.

This function is called from the external application program as follows:

hd = SapSmail(scode, sndstr, rcvstr, msg).

Description of Function Library *caddialg*

General Information on the Function Library [Seite 311]

Header Files and Libraries Required [Seite 312]

Structure of Function Calls [Seite 313]

Transfer Fields [Seite 314]

Status Information [Seite 315]

Syntax [Seite 317]

Description of Functions [Seite 318]

General functions [Seite 319]

Functions for Material Master Records [Seite 322]

Classification System Functions [Seite 323]

Functions for Document Management [Seite 324]

Functions for Plant Maintenance [Seite 325]

Functions for Engineering Change Management [Seite 326]

BOM Functions [Seite 327]

Office Functions [Seite 329]

User-Defined Functions [Seite 330]

General Rules for Data Transfer [Seite 331]

General Information on the Function Library

A special C function library is available for all hardware platforms supported by the CAD dialog interface. This function library allows you to transfer data between your external application and the SAP System, using simple sub-program calls.

This C library is the actual interface to the SAP System. The external program communicates with the SAP System by functions calls from this library.

See also:

File cadtst1.c [Seite 670]

Header Files and Libraries Required

Before you can work with the function library, header file *caddialg.h* must be installed in the application program.

See also:

File caddialg.h [Seite 653]

To set up the interface to your application, you need to install the following libraries:

- *caddialg.lib* or *caddialg.o*
- *cadrfc00.lib* or *cadrfc00.o*
- *librfcxx.lib* (xx=16 or 32) or *librfc.o*

Structure of Function Calls

The following topics contain essential information the on the structure of function calls:

Transfer Fields [Seite 314]

Status Information [Seite 315]

Syntax [Seite 317]

Transfer Fields

All transfer fields are fields of character data type. Memory is dynamically allocated to the data areas made available in each function by the SAP System.

Status Information

All function calls return a structure that consists of SAP or CPIC status information.

This structure is defined in header file *caddalg.h* under the name SAPHD, and has the following structure:

typedef struct {

char *pcode;	/*Process code	*/
char *ucode;	/* Sub-process code	*/
int subrc;	/* SAP return code	*/
int meslen;	/* Message length	*/
long datalen;	/* Data length	*/
char fkz;	/* Next record indicator	*/
long cpicrc;	/* CPI-C return code	*/
char *cpicmes;	/* CPI-C message	*/

} SAPHD;

The elements of the structure are described below:

- **pcode**

  This field contains the current process code.

- **ucode**

  This field contains the sub-process code set by the SAP System.

- **subrc**

- **subrc**

  If an error situation occurs during processing in the SAP System, the value in this field is not equal to zero.

  In this case, the SAP System usually sets an error message. No data is transferred.

- **meslen**

  This field contains the length of the message sent by the SAP System. Each message starts with the SAP error message number.

  E025 Master record already exists for < EB01 >

- **datalen**

  Length of the data sent by the SAP System.

- **fkz**

  This indicator is not equal to zero if the SAP System is holding further datasets to transfer.

  See also:

  General Rules for Data Transfer [Seite 331]

- **cpicrc**

Status Information

If the return code in this field is not equal to zero, a communication error or an error in the interface program has occurred.

This can occur in the following situations:

- No connection to the SAP System was established.
- The existing connection to the SAP System was terminated.

A return code that is not equal to zero causes the existing connection to the SAP System to be terminated. If you want to transfer more data, you need to establish a connection again. In these situations, data is no longer made available by the SAP System. For a description of the return codes, please see the appendix.

- **cpicmes**

  Short description of the error situation

  See also:

  Error messages in function library caddialg [Seite 626]

Syntax

All function calls have the same syntax.

The only exceptions are as follows: establish connection, terminate connection, query version, and terminate function.

hd = SapPcode(ucode, sndstr, rcvstr, msg)

Key:

SAPHD hd	Structure of type SAPHD
char *ucode	Sub-process code to be processed (two-digit character string)
char *sndstr	Data string to be sent (character string of any length)
char **rcvstr	Data string received from SAP System (character pointer)
char **msg	message received from SAP System in case of error (character pointer)

The data string and message received from the SAP System do not always end with hex-zero.

Description of Functions

The following topics tell you which commands you can use for the individual application areas:

General functions [Seite 319]
Functions for Material Master Records [Seite 322]
Classification System Functions [Seite 323]
Functions for Document Management [Seite 324]
Functions for Plant Maintenance [Seite 325]
Functions for Engineering Change Management [Seite 326]
BOM Functions [Seite 327]
Office Functions [Seite 329]
User-Defined Functions [Seite 330]

General functions

This topic contains information on general functions.

Initialize SAP-CAD function library for RFC connection

You must make an initialization call if an RFC connection to the R/3 System already exists, the existing connection was not established via the API SapConnc() you want to use an API call from the SAP-CAD function library

Function call:

```
hd = SapConncRfc(argv, cadsys, rcvstr, msg)
```

Key:

SAPHD hd	Structure of type SAPHD
char *argv	String containing path for starting program and program name (for example: /usr/sap/cadtst1)
char *cadsys	External system
char **rcvstr	Data string received from SAP System
char **msg	Message received from SAP System in case of error

If the call was successful, the data string received from the SAP System contains SAP status information. The structure of this data is the same as the structure of data in system function *SapSysrq*(), sub-process code "*PA*".

Establish connection to SAP System

This command is used to establish a connection to the SAP System.

Function call:

```
hd = CONNC(argv, sapsys, mandt, cadsys, user, passw,
     langu, version, msg)
```

Key:

SAPHD hd	Structure of type SAPHD
char *argv[]	Array pointer containing the program path and program name in argv[0] (Example: /usr/sap/cadtst1)
char *sapsys	SAP System (logical name)
char *mandt	SAP client
char *cadsys	External system
char *user	SAP user
char *passw	SAP user password
char *langu	SAP logon language
char *version	SAP version (R/2 or R/3)

char **msg	Message received from SAP System

All logon parameters can also be defined in the SAP configuration file *caddialg.ini,* if required. If you want to enter the parameters manually, leave the transfer parameter in file *caddialg.ini* blank.

Terminate connection to SAP System

This command is used to terminate a connection to the SAP System.

Function call:

```
hd = SapDisco()
```

Key:

SAPHD hd	Structure of type SAPHD

Terminate function

This command is used to terminate the current function at any point.

Function call:

```
hd = SapBreak(msg)
```

Key:

SAPHD hd	Structure of type SAPHD
char **msg	Message received from SAP System

Query version

This command is used to query the interface version.

Function call:

```
hd = SapVersi(version)
```

Key:

SAPHD hd	Structure of type SAPHD
char **version	Interface version number (5 bytes)

Interface synchronization

This command can be used to allocate the SAP configuration parameters and the SAP configuration tables to field sets (TCIM), defined SAP->CAD field sets, and request CAD->SAP field sets.

Function call:

```
hd = SapSysrq(ucode, sndstr, rcvstr, msg)
```

Request online help

This command supports two functions:

1. Request possible entries in SAP data field
2. Request SAP field documentation

Function call:

```
hd = SapHlprq(ucode, sndstr, rcvstr, msg)
```

Online debugging in RFC mode

This function is available as of 3.0C, (except for MS Windows 3.1).

If you use the interface in RFC mode, you can use this function call to switch on R/3 online debugging. For all subsequent API calls of the SAP-CAD function library, a SAP-GUI session is started, the SAP debugger is activated, and the program is stopped immediately before the function you want to execute.

Function call:

```
hd = SapDebug(mode)
```

Key:

SAPHD hd	Structure of type SAPHD	
char *mode	ON	Switch debugging on
	OFF	Switch debugging off

You can activate the SAP debugger by setting the environment variable SAPCAD_DEBUG.

SAPCAD_DEBUG = 1 -> switch debugging on

SAPCAD_DEBUG = 2 -> switch debugging off

Functions for Material Master Records

This topic contains information on material master record functions.

Creating a Material Master Record

Function call:

```
hd = SapMatcr(ucode, sndstr, rcvstr, msg)
```

Changing a Material Master Record

Function call:

```
hd = SapMatch(ucode, sndstr, rcvstr, msg)
```

Displaying a Material Master Record

Function call:

```
hd = SapMatrq(ucode, sndstr, rcvstr, msg)
```

Reserving a Material Number

Function call:

```
hd = SapRsrve(ucode, sndstr, rcvstr, msg)
```

Creating a Revision Level

Function call:

```
hd = SapRevcr (ucode, sndstr, rcvstr, msg)
```

Changing a Revision Level

Function call:

```
hd = SapRevch (ucode, sndstr, rcvstr, msg)
```

Displaying a Revision Level

Function call:

```
hd = SapRevrq (ucode, sndstr, rcvstr, msg)
```

Classification System Functions

This topic contains information on classification system functions.

Class list

Function call:

hd = SapSlcls(ucode, sndstr, rcvstr, msg)

Find objects in classes

Function call:

hd = SapSlobj(ucode, sndstr, rcvstr, msg)

Functions for Document Management

This topic contains information on document management functions.

Create document info record

Function call:

```
hd = SapDoccr(ucode, sndstr, rcvstr, msg)
```

Change document info record

Function call:

```
hd = SapDocch(ucode, sndstr, rcvstr, msg)
```

Display document info record

Function call:

```
hd = SapDocrq(ucode, sndstr, rcvstr, msg)
```

Check out document

Function call:

```
hd = SapDocco(ucode, sndstr, rcvstr, msg)
```

Check in document

Function call:

```
hd = SapDocci(ucode, sndstr, rcvstr, msg)
```

Create revision level

Function call:

```
hd = SapRevcr (ucode, sndstr, rcvstr, msg)
```

Change revision level

Function call:

```
hd = SapRevch (ucode, sndstr, rcvstr, msg)
```

Display revision level

Function call:

```
hd = SapRevrq (ucode, sndstr, rcvstr, msg)
```

Functions for Plant Maintenance

This topic contains information on plant maintenance functions.

Creating an Equipment Master Record

Function call:

```
hd = SapEqicr(ucode, sndstr, rcvstr, msg)
```

Changing an Equipment Master Record

Function call:

```
hd = SapEqich(ucode, sndstr, rcvstr, msg)
```

Displaying an Equipment Master Record

Function call:

```
hd = SapEqirq(ucode, sndstr, rcvstr, msg)
```

Creating a Functional Location

Function call:

```
hd = SapTpccr(ucode, sndstr, rcvstr, msg)
```

Changing a Functional Location

Function call:

```
hd = SapTpcch(ucode, sndstr, rcvstr, msg)
```

Displaying a Functional Location

Function call:

```
hd = SapTpcrq(ucode, sndstr, rcvstr, msg)
```

Creating a Maintenance Notification

Function call:

```
hd = SapPmncr(ucode, sndstr, rcvstr, msg)
```

Functions for Engineering Change Management

This topic contains information on engineering change management functions.

Create change master record

Function call:

```
hd = SapChgcr(ucode, sndstr, rcvstr, msg)
```

Change change master record

Function call:

```
hd = SapChgch(ucode, sndstr, rcvstr, msg)
```

Display change master record

Function call:

```
hd = SapChgrq(ucode, sndstr, rcvstr, msg)
```

Create ECR

Function call:

```
hd = SapEcrcr(ucode, sndstr, rcvstr, msg)
```

Change ECR

Function call:

```
hd = SapEcrch(ucode, sndstr, rcvstr, msg)
```

Display ECR

Function call:

```
hd = SapEcrrq(ucode, sndstr, rcvstr, msg)
```

Functions for BOMs

This topic contains information on BOM functions.

Create BOM

Function call:

hd = SapBomcr(ucode, sndstr, rcvstr, msg)

Create BOM With Sub-Items

Function call:

```
rc = CadBomCreateSubItem
(BILSK                          * MaterialHeaderData,
CadRfcImportField       * ImportData,
ITAB_H                          ** SapFieldData,
ITAB_H                          ** DmsClassData);
```

Change BOM

Function call:

hd = SapBomcn(ucode, sndstr, rcvstr, msg)

Change BOM With Sub-Items

Function call:

Reading and locking the current BOM

```
rc= CadBomChangeSubItemRead
(BICSK                          * MaterialHeaderData,
CadRfcImportField       * ImportData,
ITAB_H                          ** SapFieldData,
ITAB_H                          ** DmsClassData);
```

Making the BOM change

```
rc= CadBomChangeSubItem
(BICSK                          * MaterialHeaderData,
CadRfcImportField       * ImportData,
ITAB_H                          ** SapFieldData,
ITAB_H                          ** DmsClassData);
```

Display BOM

Function call:

hd = SapBomrq(ucode, sndstr, rcvstr, msg)

Functions for BOMs

Display BOM With Sub-Items

Function call:

```
rc= CadBomDisplaySubItem
(BICSK                          * MaterialHeaderData,
CadRfcImportField       * ImportData,
ITAB_H                          ** SapFieldData,
ITAB_H                          ** DmsClassData);
```

Where-used list

Function call:

```
hd = SapBomwu(ucode, sndstr, rcvstr, msg)
```

Office Functions

This topic contains information on Office functions.

Function call:

hd = SapSmail(ucode, sndstr, rcvstr, msg)

User-Defined Functions

This topic contains information on user-defined functions.

A total of 5 function calls are supported:

Function call syntax:

```
hd = SapUsr01(ucode, sndstr, rcvstr, msg)
hd = SapUsr02(ucode, sndstr, rcvstr, msg)
hd = SapUsr03(ucode, sndstr, rcvstr, msg)
hd = SapUsr04(ucode, sndstr, rcvstr, msg)
hd = SapUsr05(ucode, sndstr, rcvstr, msg)
```

General Rules for Data Transfer

To ensure that data is transferred successfully, please note the following rules:

- After each function call you do, always query the CPIC return code. The dialog step was only successful if the return code is zero. Only then can you assume that the SAP data fields have been filled out. If the return code is not zero, there is usually an explanation of the error that occurred in field *cpicmes*.
- If large quantities of data is transferred (for example, BOMs or classification data), the data must not be split into several part blocks if the maximum block size (currently 25000 bytes) is exceeded.

  This task is performed by the SAP function. This means that all the data to be sent or received should be made available, even in the case of long datasets. The next record indicator in the data header is only required if several blocks of data that have different sub-process codes are transferred.
- In various functions, such as the function for displaying material master records with classification data, the SAP System sends datasets with different sub-process codes one after the other. To receive these subsequent datasets, you need to repeat the function call. You can put any data in the string to be sent. The important thing is to ensure that the sub-process code is reset to hex 0 or space for these repeat function calls.

Data transfer: Example

You request information on a material master record. You also want to see the classification data. To do this, you use library function MATRQ with sub-process code MC.

```
hd = SapMatrq("MC", (char *) sndstr, (char **) rcvstr, (char **) msg)
```

After the first function call, the data area referred to by the rcvstr pointer contains the actual material master data. The next record indicator hd.fkz is not blank. This means that the function called is not active and that the SAP System is holding further data. To see data on the class to which the material master is allocated, call the function again without the sub-process code.

```
hd = SapMatrq("", (char *) sndstr, (char **) rcvstr, (char **) msg)
```

The next record indicator hd.fkz is still not blank. The third time you call the function, you see the classification data for the material master record.

```
hd = SapMatrq("", (char *) sndstr, (char **) rcvstr, (char **) msg)
```

The function is now completed and the next record indicator is deleted.

You cannot call a new function until the next record indicator is blank. However, you can interrupt any function you have started at any time with the BREAK function.

Functions of the CAD Toolbox Library

This section describes the functions of the CAD toolbox library.

Introduction [Seite 333]

Installation and Implementation [Seite 334]

Functions [Seite 335]

Appendix [Seite 412]

Introduction

This document is intended for anyone who is developing new C applications or converting existing C applications to run with the SAP-CAD libraries.

Benefits

Using this toolbox, you can integrate the SAP function library in your own application much more quickly. This section assumes an understanding of the functions in the SAP Function Library.

This section shows you how to write C programs that use the SAP Function Library and the SAP Toolbox Library to access and manipulate SAP data. It provides an easy-to-follow example, instructions, and programming tips to improve your understanding of the SAP-CAD libraries.

Before you start

To use this section effectively, you need a working knowledge of the following:

- applications programming in C
- concepts and terminology of the SAP-CAD interface

Installation and Implementation

All functions of the toolbox return integer values. The datasets that are output by the functions are allocated dynamically within the functions. The output should be used for read only purposes.

If the toolbox recognizes an error, a return value not equal to zero is returned. To see the description of the message, use function **SapGetErrorMessage**.

> Please ensure that you call the functions **SapInitToolbox** (see: Init Toolbox [Seite 338]) and **SapFillInterface** (see: Save Configuration Parameters [Seite 342]) before you use other toolbox functions.

Functions that can be used within the toolbox are described in this section. Examples of communication procedures are shown in C program *CADSAMPL.C*.

The toolbox requires two configuration files to work.

- *CADTMES.DAT*

  CADTMES.DAT contains descriptions of messages.

- *CADTTAB.DAT*

  CADTTAB:DAT is required for R/2 classification functions.

These files must be installed when you install the application. To see the location of these files, use the function *SapConfigureToolbox* (see: Configure Toolbox [Seite 337]).

For more information on files *CADTMES.DAT* and *CADTTAB.DAT*, see the following topics:

CADTMES.DAT [Seite 414]

CADTTAB.DAT [Seite 418]

Internal structures within the toolbox

You can access several internal structures within the toolbox. The parameter that you enter when you call a function determines which structure is to be processed. In this function description, the following are among the keys used for these structures (internal tables):

- U1 table with upload data
- D1 table with download data
- U2 other tables for upload data that can be maintained in parallel with U1 (similarly U3 and D2)

Functions

Functions for the following application areas are supported in the CAD toolbox library:

General functions [Seite 336]

Utility Functions [Seite 344]

Master Record Functions [Seite 349]

Classification Functions [Seite 362]

Match Code Functions [Seite 389]

Online Help Functions [Seite 395]

Mail Functions (R/3 2.1/2.2) [Seite 402]

User-Defined Functions [Seite 410]

General Functions

The following general functions are supported in the CAD toolbox library:

Configure Toolbox [Seite 337]

Init Toolbox [Seite 338]

Load Configuration Data [Seite 339]

Load Error Messages [Seite 340]

Load Toolbox Version [Seite 341]

Save Configuration Parameters [Seite 342]

Save Configuration Tables [Seite 343]

Configure Toolbox

With this function, you can configure the toolbox. You can change the standard configuration for the class description and the message files (CADTMES.DAT and CADTTAB.DAT).

intSapConfigureToolbox (order,filename)

Input/Output	Parameter	Type	Description
Input	order	int	1 - name of message file 2 - name of class description file
	filename	char*	new name and path of the file

Init Toolbox

With this function, you initialize the toolbox at the beginning of your main program.

intSapInitToolbox (order)

Input/Output	Parameter	Type	Description
Input	order	int	1 - system start 2 - initialization of buffers

This function must be called at the beginning of the application program!

Load Configuration Data

This function loads information about the current configuration of the SAP-CAD Interface from the toolbox memory.

intSapDescribeInterface (order,string)

Input/Output	Parameter	Type	Description
Input	order	int	1 - separator 2 - release 3 - matchcode identifier 4 - matchcode separator 5 - date format 6 - decimal point
Output	string	char**	value received

See also:

Save Configuration Parameters [Seite 342]

Load Error Message

With this function, you can load a description for an error code. All error codes are stored in file *CADTMES.DAT*.

intSapGetErrorMessage (mesnum,messtr)

Input/Output	Parameter	Type	Description
Input	mesnum	int	message number
Output	messtr	SAPMES*	Message structure SAPMES [Seite 435]

Load Version of Toolbox

This function loads the current version of the toolbox.

intSapToolboxVersion (version)

Input/Output	Parameter	Type	Description
Output	version	char**	version of the toolbox

Save Configuration Parameters

This function saves the configuration data (separator, matchcode sign, and so on) to the toolbox. Configuration data of the interface is described in SAP table TCIS and returned by function SapSysrq("PA"...) as a receive string. You must call this function before you work with the toolbox.

intSapFillInterface (rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	rcvstr	char*	data received
	rcvlen	int	length of data received

See also:

Load Configuration Data [Seite 339]

Save Configuration Tables

This function saves the current configuration of the SAP tables TCIU, TCID, TCIM to local ASCII files. The functions SapSysrq("TB"...) return one of the above files as a receive string. The toolbox needs this files to set up its internal tables and send strings.

intSapWriteKonfiguration (order,rcvstr,rcvlen,filename)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table with upload data (TCIU) 2 - table with download data (TCIM) 3 - table with defined sets (TCIM)
	rcvstr	char*	data received from SAP
	rcvlen	int	length of data received
	filename	char*	name of ASCII file for storing the SAP table

See also:

Create Internal Table [Seite 351]

Create and Fill Internal Table [Seite 352]

Load Attribute Value for Record Number [Seite 357]

Load Attribute Value for Attribute Name [Seite 358]

SaveTable Values [Seite 361]

Load Attribute Name [Seite 346]

Find Attribute [Seite 348]

Utility Functions

The following utilities are supported by the CAD toolbox library:

Delete Table Entry [Seite 345]
Load Attribute Name [Seite 346]
Print Internal Table [Seite 347]
Find Attribute [Seite 348]

Delete Table Entry

This function allows you to delete existing table entries.

intSapDeleteTableEntry (order,tab_line)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3) 10 - table with user-defined data
	tab_line	int	number of lines to delete

Load Attribute Name

This function loads an attribute name from the configuration files CIU or CID. The attribute name is read for a specified function step (process code and sub-process code). Parameter INDEX describes the number of the attribute within the field set for this function step.

intSapGetAttributName (order,pcode,ucode,cimnam,filename,index,attstruct)

Input/Output	Parameter	Type	Description
Input	order	int	1 - read table TCIU 2 - read table TCID
	pcode	char*	Process code
	ucode	char*	Sub-process code
	cimnam	char*	name of configuration file CIM
	datnam	char*	name of configuration file CIU/CID
	index	int	attribute number
Output	attstruct	SAPATT*	Attribute structure SAPATT [Seite 436]

See also:

Find Attribute [Seite 348]

Print Internal Table

This function stores the contents of the specified internal table in an ASCII file.

intSapPrintTable (order,filename)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3) 10 - table with user-defined data 1 - 11 - table with class list (R/3 upload) 12 - table for class list (R/3 download) 21 - table with mailbox data (R/3 2.2) 22 - table with mail text data (R/3 2.2) 23 - table with mail header data (R/3 2.2) 24 - table with mail parameter 1 (R/3 2.2) 25 - table with mail parameter 2 (R/3 2.2) 26 - table with mail recipient list (R/3 2.2)
	filename	char*	name of configuration file

If an internal table was created with order 6, you can use function SapPrintTable to output the table to an ASCII file.

Find Attribute

This function finds the attribute name entered in the local configuration files. If the attribute name does not exist, the function returns 58.

intSapSearchAttribut (order,pcode,ucode,cimnam,datnam,attnam,attstr)

Input/Output	Parameter	Type	Description
Input	order	int	1 - read table TCIU 2 - read table TCID
	pcode	char*	Process code
	ucode	char*	Sub-process code
	cimnam	char*	name of configuration file CIM
	datnam	char*	name of configuration file CIU/CID
	attnam	char*	name of configuration attribut
Output	attstr	SAPATT*	Attribute structure SAPATT [Seite 436]

See also:

Load Attribute Name [Seite 346]

Master Record Functions

All the functions described in the topics below can be used with SAP objects, such as material master records, document info records, equipment master records, bills of material, and so on.

The following master record functions are supported by the CAD toolbox library:

Count Table Entries (Master Record Functions) [Seite 350]

Create Internal Table [Seite 351]

Create and Fill Internal Table [Seite 352]

Create Send String [Seite 354]

Create Send String for BOM Header Data (R/3) [Seite 355]

Fill Internal Table [Seite 356]

Load Attribute Value for Record Number [Seite 357]

Load Attribute Value for Attribute Name [Seite 358]

Change Table Value [Seite 360]

SaveTable Values [Seite 361]

Count Table Entries (Master Record Functions)

This function computes the number of entries (records) in the table you enter.

intSapDescribeTable (order,string)

Input/Output	Parameter	Type	Description
			1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3) 10 - table with user-defined data
Output	string	string**	number of entries

The order parameter must have the value 1 if you want the number of entries in an internal table created with order 6 to be output.

See also:

Create Internal Table [Seite 351]

Create Characteristics Data Table [Seite 374]

Create and Fill Internal Table [Seite 352]

Create and Fill Matchcode Table [Seite 391]

Create Internal Table

This function is used to create an internal table for downloading or uploading data. The structure of the internal tables is described in the configuration files CIM, CIU and CID.

intSapCreateTable (order,pcode,ucode,cimnam,datnam)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3) 10 - table with user-defined data
	pcode	char*	Process code
	ucode	char*	Sub-process code
	cimnam	char*	name of configuration file CIM
	datnam	char*	name of configuration file CIU/CID

If an internal table was created with order 6, the value 1 must be entered as the order for function SapDescribeTable, SapPrintTable, and SapGetTableValue

See also:

Create and Fill Internal Table [Seite 352]

Load Attribute Name [Seite 346]

Load Attribute Value for Record Number [Seite 357]

Load Attribute Value for Attribute Name [Seite 358]

Change Table Value [Seite 360]

Find Attribute [Seite 348]

SaveTable Values [Seite 361]

Save Matchcode Field Value [Seite 394]

Create and Fill Internal Table

This function is used to create and fill an internal table.

For this function, the data string received by the CAD interface function (for example, a string containing material master data from the SAP System) is transferred as parameter *rcvstr.* The parameter *recvlen* defines the length of the data string (hd.datalen). This function generates an internal structure with attribute name and values.

intSapCreateFillTable (order,pcode,ucode,cimnam,datnam,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3) Orders 7 and 8 allow you to keep two upload tables in memory in parallel.
	pcode	char*	Process code
	ucode	char*	Sub-process code
	cimnam	char*	name of configuration file CIM
	datnam	char*	name of configuration file CIU/CID
	rcvstr	char*	data received
	rcvlen	long	length of data received

If an internal table was created with order 6, the value 1 must be entered as the order for function SapDescribeTable, SapPrintTable, and SapGetTableValue

See also:

Create Internal Table [Seite 351]

Load Attribute Name [Seite 346]

Load Attribute Value for Record Number [Seite 357]

Load Attribute Value for Attribute Name [Seite 358]

Change Table Value [Seite 360]

Find Attribute [Seite 348]

Save Configuration Tables [Seite 343]

SaveTable Values [Seite 361]

Save Matchcode Field Value [Seite 394]

Create Send String

This function is used to create a send string containing the data described in configuration file CIU. The function allocates memory for the string dynamically.

intSapCreateSendstr (sendstr)

Input/Output	Parameter	Type	Description
Output	datstr	char**	send string for upload

See also:

Create Internal Table [Seite 351]

Create and Fill Internal Table [Seite 352]

Fill Internal Table [Seite 356]

Create Send String for BOM Header Data (R/3)

This function is used to create a send string containing BOM header data. Two commands are required to save old and new header data to different internal tables. This makes it possible to access old and new data at the same time.

intSapCreateSendstrBomHead (order, sendstr)

Input/Output	Parameter	Type	Description
Input	order	int	7 - table with new BOM header data 8 - table with old BOM header data
Output	sendstr	char**	Send string for upload to SAP

See also:

Create Internal Table [Seite 351]

Fill Internal Table [Seite 356]

Fill Internal Table

This function is used to insert data in an internal table. The internal table structure must already exist before you call this function:

intSapFillTable (order,pcode,ucode,cimnam,datnam,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3)
	rcvstr	char*	data received
	rcvlen	long	length of data received

See also:

Create Internal Table [Seite 351]

Load Attribute Name [Seite 346]

Load Attribute Value for Record Number [Seite 357]

Load Attribute Value for Attribute Name [Seite 358]

Change Table Value [Seite 360]

Find Attribute [Seite 348]

Save Configuration Tables [Seite 343]

SaveTable Values [Seite 361]

Save Matchcode Field Value [Seite 394]

Load Attribute Value for Record Number

This function reads entries from an internal table. The data record number that you enter acts as an access key.

The attribute position determines the number of the attributes to be read. Value_no determines the value of the attribute to be read (value_no is usually 1).

intSapGetAttributNameValue (order,position,value_no,attnam,value)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3)
	position	int	position of attribute (record number)
	value_no	int	number of attribute value
Output	attnam	char*	attribute name
	value	char**	attribute value

See also:

Create Internal Table [Seite 351]

Create and Fill Internal Table [Seite 352]

Load Attribute Name [Seite 346]

Load Attribute Value for Attribute Name [Seite 358]

Change Table Value [Seite 360]

Find Attribute [Seite 348]

Save Configuration Tables [Seite 343]

SaveTable Values [Seite 361]

Save Matchcode Field Value [Seite 394]

Load Attribute Value for Attribute Name

This function reads entries from an internal table. It uses the attribute name you enter as an access key and the attribute value_no is the number of the attribute value. Several values can exist for one attribute name.

intSapGetTableValue (order,attnam,value_no,value)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3) 10 - table with user-defined data Order 5 returns the value of the attribute in table D1. If the attribute does not exist in table D1, order 5 returns the value of the attribute in table D2.
	attnam	char*	attribute name
	value_no	int	number of attribute value
Output	value	char**	attribute value

If an internal table was created with order 6, the value 1 must be entered as the order for function SapDescribeTable, SapPrintTable, and SapGetTableValue

See also:

Create Internal Table [Seite 351]

Create and Fill Internal Table [Seite 352]

Load Attribute Name [Seite 346]

Load Attribute Value for Record Number [Seite 357]

Change Table Value [Seite 360]

Find Attribute [Seite 348]

Save Configuration Tables [Seite 343]

SaveTable Values [Seite 361]

Save Matchcode Field Value [Seite 394]

Change Table Value

This function changes the value of the attribute you enter in an internal table.

intSapModifyTableValue (order,attnam,value,value_no)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table with upload data 2 - table with download data 7 - table with upload data (new BOM header) 8 - table with upload data (old BOM header)
	attnam	char*	attribute name
	value	char*	attribute value
	value_no	int	number of attribute value

See also:

Create Internal Table [Seite 351]

Create and Fill Internal Table [Seite 352]

Load Attribute Name [Seite 346]

Load Attribute Value for Record Number [Seite 357]

Load Attribute Value for Attribute Name [Seite 358]

Find Attribute [Seite 348]

SaveTable Values [Seite 361]

Save Table Value

This function saves the value of the attribute you enter to an internal table.

intSapWriteTableValue (order,attam,value,value_no)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload (U1) 2 - download (D1) 3 - download (matchcode) 4 - download classification data (R/2) 5 - download (D2) 6 - upload classification data (R/2) 7 - upload (U2) 8 - upload (U3) 10 - table with user-defined data
	attnam	char*	attribute name
	value	char*	attribute value
	value_no	int	number of attribute value

See also:

Create Internal Table [Seite 351]

Create and Fill Internal Table [Seite 352]

Change Table Value [Seite 360]

Classification Functions

The following classification functions are supported by the CAD toolbox library:

Count Table Entries (Classification Functions) [Seite 363]
Count Characteristic Values (R/3) [Seite 364]
Create Send String for Allowed Characteristic Values (R/3) [Seite 365]
Create Send String for Characteristics Data (R/2 and R/3) [Seite 366]
Create Send String for Class Data (R/3) [Seite 367]
Create Send String for Class Scope (R/3) [Seite 368]
Create Send String for Finding Classes (R/3) [Seite 369]
Create Send String for Finding Objects via Classes (R/3) [Seite 370]
Create and Fill Class Hierarchy Table (R/3) [Seite 371]
Create and Fill Classification Table (R/3) [Seite 372]
Create and Fill Characteristic Values Table (R/3) [Seite 373]
Create Characteristics Data Table [Seite 374]
Delete Search Result (R/2) [Seite 375]
Load Classification Data for Characteristic Number (R/2) [Seite 376]
Load Classification Data for Characteristic Name (R/2) [Seite 377]
Load Classification Data for Characteristic Number (R/3) [Seite 378]
Load Characteristic Value Data (R/3) [Seite 380]
Load Object Data (R/3) [Seite 381]
Load Class Data (R/3) [Seite 382]
Load Class Hierarchy Data (R/3) [Seite 383]
Load Search Result Data (R/3) [Seite 384]
Save Characteristic Value and Description [Seite 385]
Save Search Result [Seite 388]

Count Table Entries (Classification Functions)

This function returns a description of the internal object table. This table contains the list of objects produced by the *Find objects* function from the SAP-CAD library. If you are working with classification data for an SAP object, use the *SapDescribeTable* function (see: Count Table Entries (Master Record Functions) [Seite 350]) to determine the number of characteristics.

intSapGetSmlDescription (order,string)

Input/Output	Parameter	Type	Description
Input	order	int	1 - number of characteristics 2 - number of objects 3 - class name 4 - class type 5 - class table (R/2) class description (R/3) 6 - class catchword (R/2) class status text (R/3) 7 - class structure name (R/2) indicator: standard class (R/3)
Output	string	char**	Value

See also:

Create Send String for Characteristics Data (R/2 and R/3) [Seite 366]

Create Characteristics Data Table [Seite 374]

Delete Search Result (R/2) [Seite 375]

Load Classification Data for Characteristic Number (R/2) [Seite 376]

Load Classification Data for Characteristic Number (R/3) [Seite 378]

Load Classification Data for Characteristic Name (R/2) [Seite 377]

Load Classification Data for Characteristic Name (R/3) [Seite 379]

Save Characteristic Value and Description [Seite 385]

Count Characteristic Values (R/3)

This function returns the number of the values for the characteristic you enter.

intSapGetSmlValueDescriptionR3 (order, obj_no, char_name, number)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table for finding objects via classes 2 - table of classification data of an object 3 - table of allowed characteristic values
	obj_no	int	object number
	char_name	char*	name of the characteristic
Output	number	int*	number of values

Create Send String for Allowed Characteristic Values (R/3)

This function is used to create a send string containing a list of characteristics. The characteristics are read either from the internal table containing the search result of a *Find objects via classes* function, or from the internal table containing the characteristics of one specific object.

intSapCreateSendstrPosSmlValuesR3 (order, sendstr)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table for finding objects via classes 2 - table for classification data of an object
Output	sendstr	char**	send string for upload

Create Send String for Characteristics Data (R/2 and R/3)

This function is used to create a send string for the classification system. This string is generated from an existing internal table. The function allocates memory for the string dynamically.

intSapCreateSendstrSml (order,sendstr)

Input/Output	Parameter	Type	Description
Input	order	int	1 - send string for finding objects via classes 2 - send string for classification data of an object
Output	sendstr	char**	send string for upload

See also:

Create Characteristics Data Table [Seite 374]

Delete Search Result (R/2) [Seite 375]

Save Search Result [Seite 388]

Create Send String for Class Data (R/3)

This function is used to create a send string with information about one specific class.

intSapCreateSendstrClasR3 (type, name, m_oblig, m_optional, m_eval, m_not_eval, m_inherited, m_std_class, sendstr)

Input/Output	Parameter	Type	Description
Input	type	char*	Class type
	name	char*	Class name
	m_oblig	char*	Indicator: required characteristics
	m_optional	char*	Indicator: optional characteristics
	m_eval	char*	Indicator: characteristics with values
	m_not_eval	char*	Indicator: characteristics without values
	m_inherited	char*	Indicator: inherited characteristics
	m_std_class	char*	Indicator: standard class
Output	sendstr	char**	Send string for upload to SAP

Create Send String for Class Scope (R/3)

Define scope for class

This function is used to create a send string containing the scope you want to define in the SAP System.

intSapCreateSendstrClasScopeR3 (type, m_oblig, m_optional, m_eval, m_not_eval, m_inherited, sendstr)

Input/Output	Parameter	Type	Description
Input	type	char*	Class type
	m_oblig	char*	Indicator: required characteristics
	m_optional	char*	Indicator: optional characteristics
	m_eval	char*	Indicator: characteristics with values
	m_not_eval	char*	Indicator: characteristics without values
	m_inherited	cha*	Indicator: inherited characteristics
Output	sendstr	char**	Send string for upload to SAP

Define new scope for finding objects via classes

This function is used to create a send string with the new scope for class selection.

intSapCreateSendstrClasSelScopeR3 (m_oblig, m_optional, m_eval,m_not_eval, m_inherited, m_obj_not_eval, m_obj_locked, m_obj_assigned)

Input/Output	Parameter	Type	Description
Input	m_oblig	char*	Indicator: required characteristics
	m_optional	char*	Indicator: optional characteristics
	m_eval	char*	Indicator: characteristics with values
	m_not_eval	char*	Indicator: characteristics without values
	m_inherited	char*	Indicator: inherited characteristics
	m_obj_not_eval	char*	Indicator: object not evaluated
	m_obj_locked	char*	Indicator: object locked
	m_obj_assigned	char*	Indicator: object assigned
Output	sendstr	char**	Send string for upload to SAP

Create Send String for Finding Classes(R/3)

This function is used to create a send string for a specific class and a specific scope. The output string is used to call function SapSlobj("RI"...).

intSapCreateSendstrClasSearchR3 (order, type, name, date, langu, m_assigned, upper_level, lower_level, datstr)

Input/Output	Parameter	Type	Description
Input	order	int	1 - find classes generically 2 - find classes without superior classes 3 - find superior and subordinate classes
	type	char*	Class type
	name	char*	Class name
	date	char*	date
	langu	char*	language
	m_assigned	char*	Indicator: with objects allocated
	upper_level	char*	number of levels upward
	lower_level	char*	number of levels downward
Output	datstr	char**	Send string for upload to SAP

Create Send String for Finding Objects via Classes (R/3)

This function is used to create a send string with information about one specific class.

intSapCreateSendstrClasSelectR3 (type, name,date, m_oblig, m_optional, m_eval, m_not_eval, m_inherited, m_obj_not_eval, m_obj_locked, m_obj_assigned, datstr)

Input/Output	Parameter	Type	Description
Input	type	char*	Class type
	name	char*	Class name
	date	char*	date
	m_oblig	char*	Indicator: required characteristics
	m_optional	char*	Indicator: optional characteristics
	m_eval	char*	Indicator: characteristics with values
	m_not_eval	char*	Indicator: characteristics without values
	m_inherited	cha*	Indicator: inherited characteristics
	m_obj_not_eval	char*	Indicator: objects without characteristic values
	m_obj_locked	char*	Indicator: locked objects
	m_obj_assigned	char*	Indicator: allocated objects
Output	datstr	char**	Send string for upload to SAP

Create and Fill Class Hierarchy Table(R/3)

This function is used to create an internal table containing the class hierarchy and fill this table. The receive sting is the output of the SAP function SapSlcls(“RH”,...).

intSapCreateFillClasHierarchTableR3 (order,rcvstr,rcvlen,clasnum)

Input/Output	Parameter	Type	Description
Input	order	int	1 - Class hierarchy
	rcvstr	char*	data string from SAP System
	recvlen	long	length of data string
Output	clasnum	int*	number of classes in table

Create and Fill Classification Table (R/3)

This function is used to create a table for a list of classes and fill this table.

intSapCreateFillClasTableR3 (order,rcvstr,rcvlen,clasnum)

Input/Output	Parameter	Type	Description
Input	order	int	1 - upload table for class list 2 - download table for class list 3 - class list (find generically) 4 - class list (find via matchcode)
	rcvstr	char*	data string from SAP System
	rcvlen	long	length of data string
Output	clasnum	int*	number of classes in table

Create and Fill Characteristic Values Table(R/3)

intSapCreateFillPosSmlValueTableR3 (order,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - Create table of allowed characteristic values
	rcvstr	char*	data string from SAP System
	recvlen	long	length of data string

Create Characteristics Data Table

This function is used to create a table for characteristics data.

intSapCreateSmlTable (order,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table for finding objects via classes 2 - table of classification data for an object
	rcvstr	char*	data string from SAP System
	rcvlen	int	length of data string

See also:

Create Characteristics Data Table [Seite 374]

Delete Search Result (R/2) [Seite 375]

Save Search Result [Seite 388]

Load Classification Data for Characteristic Number (R/2) [Seite 376]

Load Classification Data for Characteristic Number (R/3) [Seite 378]

Load Classification Data for Characteristic Name (R/2) [Seite 377]

Load Classification Data for Characteristic Name (R/3) [Seite 379]

Count Table Entries (Classification Functions) [Seite 363]

Save Characteristic Value and Description [Seite 385]

Delete Search Result (R/2)

This function deletes the search result that is currently stored in the internal classification structure. If you want to start a new search in the SAP System, you must delete the previous search result before you can save the search result.

intSapDeleteSmlSelection()

See also:

Create Characteristics Data Table [Seite 374]

Save Search Result [Seite 388]

Load Classification Data for Characteristic Number (R/2)

This function loads the value from the internal classification table for a specified object, using the characteristic number you enter as an access key.

intSapGetSmlData (order,position,obj_no,char_data)

Input/Output	Parameter	Type	Description
Input	order	int	1 - characteristics table 2 - table with matchcode data 3 - mailbox data table (R/3 2.2) 4 - mail text table (R/3 2.2) 5 - mail header table (R/3 2.2) 6 - table with mail parameter 1 (R/3 2.2) 7 - table with mail parameter 2 (R/3 2.2) 8 - recipient list table (R/3 2.2)
	position	int	characteristic number
	obj_no	int	Object number
Output	char_data	SAPSML*	Characteristic structure SAPSML [Extern]

See also:

Create Characteristics Data Table [Seite 374]

Delete Search Result (R/2) [Seite 375]

Load Classification Data for Characteristic Name (R/2) [Seite 377]

Count Table Entries (Classification Functions) [Seite 363]

Save Characteristic Value and Description [Seite 385]

Load Classification Data for Characteristic Name (R/2)

This function loads the value from the internal classification table for a specified object, using the characteristic name or description you enter as an access key.

intSapGetSmlAttributData (order,char_nam,obj_no,char_data)

Input/Output	Parameter	Type	Description
Input	order	int	1 - key = characteristic description 2 - key = characteristic field name
	char_nam	char*	Name of the characteristic
	obj_no	int	Object number
Output	char_data	SAPSML*	Characteristic structure SAPSML [Extern]

See also:

Create Characteristics Data Table [Seite 374]

Delete Search Result (R/2) [Seite 375]

Load Classification Data for Characteristic Number (R/2) [Seite 376]

Count Table Entries (Classification Functions) [Seite 363]

Save Characteristic Value and Description [Seite 385]

Load Classification Data for Characteristic Number (R/3)

This function loads the value from the internal classification table for a specified object, using the characteristic number you enter as an access key.

intSapGetSmlDataR3 (order,position,obj_no,char_data)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table for finding objects via classes 2 - table of classification data for an object
	position	int	characteristic number
	obj_no	int	Object number
Output	char_data	SAPSML_R3	Characteristic structure SAPSML_R3 [Seite 430]

See also:

Create Characteristics Data Table [Seite 374]

Load Classification Data for Characteristic Name (R/3) [Seite 379]

Count Table Entries (Classification Functions) [Seite 363]

Save Characteristic Value and Description [Seite 385]

Load Classification Data for Characteristic Name (R/3)

This function loads the value from the internal classification table for a specified object, using the characteristic name or description you enter as an access key.

intSapGetSmlAttributDataR3 (order,char_nam,obj_no,char_data)

Input/Output	Parameter	Type	Description
Input	order	int	1 - key = characteristic name 2 - key = characteristic description
	char_nam	char*	Characteristic name
	obj_no	int	Object number
Output	char_data	SAPSML_R3*	Characteristic structure SAPSML_R3 [Seite 430]

See also:

Create Characteristics Data Table [Seite 374]

Load Classification Data for Characteristic Number (R/3) [Seite 378]

Count Table Entries (Classification Functions) [Seite 363]

Save Characteristic Value and Description [Seite 385]

Load Characteristic Value Data (R/3)

This function returns the value of the characteristic you enter. You can access multiple values of a characteristic. You can enter either the name or the number of the characteristic. If you want to enter a characteristic number, the parameter CHAR_NAME must be “NULL” or “ ” (blank character).

intSapGetSmlDataValueR3 (order,char_name,char_no,obj_no,value_no,value)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table for finding objects via classes 2 - table of classification data for an object
	char_name	char*	Characteristic name
	char_no	int	characteristic number
	obj_no	int	Object number
	valule_no	int	value number
Output	value	char**	Value

See also:

Create Characteristics Data Table [Seite 374]

Load Classification Data for Characteristic Number (R/3) [Seite 378]

Load Classification Data for Characteristic Name (R/3) [Seite 379]

Load Object Oata (R/3)

This function returns a structure containing object data. The specified object is read from the internal table that contains the list of objects returned by the *Find objects via classes* function.

intSapGetObjectDataR3 (order, object_no, obj_data)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table of selected objects
	object_no	int	number of object in internal table
Output	obj_data	SAPOBJ_R3*	Structure with object data SAPOBJ_R3 [Seite 429]

See also:

Load Characteristic Value Data (R/3) [Seite 380]

Count Characteristic Values (R/3) [Seite 364]

Load Class Data (R/3)

This function returns a structure containing class data for one specific class. This class is read from the internal table that contains a list of classes.

intSapGetClasDataR3 (order,class_no, class_data)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table of class upload data 2 - talbe of class download data
	class_no	int	number of class in internal table
Output	class_data	SAPCLAS_R3*	Structure with class data SAPCLAS_R3 [Seite 432]

See also:

Create and Fill Classification Table (R/3) [Seite 372]

Load Class Hierarchy Data (R/3)

This function returns a structure containing class hierarchy data for one specific class. This class is read from the internal table that contains the class hierarchy.

intSapGetClasHierarchDataR3 (order,class_no,class_hier_data)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table with hierarchy class data
	class_no	int	number of class in internal table
Output	class_hier_data	SAPCLAS_H_R3*	Structure with class hierarchy data SAPCLAS_H_R3 [Seite 434]

See also:

Create and Fill Class Hierarchy Table (R/3) [Seite 371]

Load Search Result Data (R/3)

This function returns a structure with search result data for one specific class. This class is read from the internal table that contains the result of the functions for finding objects via classes.

intSapGetClasSearchDataR3 (order, class_no, class_data)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table with class search result data
	class_no	int	number of class in internal table
Output	class_data	SAPCLAS_S_R3*	structure with class search result data SAPCLAS_S_R3 [Seite 433]

See also:

Create and Fill Classification Table (R/3) [Seite 372]

Save Characteristic Value and Description

This function saves the value and description of a characteristic to the internal characteristics table. You can change either an entry with a specific position number, or a characteristic with a specific name. If the parameter CHAR_NO is not equal to 0, the record with the specified CHAR_NAME is updated.

If the parameter VALUE_NO is not equal to 0, the value is added to the existing values of the characteristic.

intSapWriteSmlValue (order,char_no,char_name,field_no,value,value_no)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table of characteristics data 2 - table of matchcode data (R/2) 3 - internal use (R/2) 4 - internal use (R/2)
	char_no	int	characteristic number
	char_name	char*	Characteristic name

Save Characteristic Value and Description

	field_no	int	number of field to update **R/2:** 1 - name 2 - structure name 3 - DIN code letter 4 - input restrictions 5 - unit 6 - format 7 - template 8 - length 9 - number of decimal places 10 - characteristic value 11 - selection criteria 12 - comparison type (INCL, EXCL, MASK...) **R/3:** 1 - name 2 - description 3 - indicator: required characteristic 4 - indicator: inherited 5 - indicator: with values 6 - data type 7 - indicator: single-value 8 - indicator: intervals allowed 9 - indicator: additional values allowed 10 - length 11 - number of decimal places 12 - indicator: negative values allowed 13 - exponent display on screen 14 - exponent display 15 - template 16 - indicator: capitalized/small letters 17 - external unit of measure 18 - code letter field 19 - document type 20 - document number 21 - document part 22 - document version 23 - indicator: view 0 24 - indicator: view 1 25 - indicator: view 2 26 - indicator: view 3 27 - indicator: view 4 28 - indicator: view 5 29 - indicator: view 6 30 - indicator: view 7 31 - indicator: view 8 32 - indicator: view 9 33 - characteristic value 34 - selection criteria 35 - comparison type (INCL, EXCL, MASK...)
	value	char*	Characteristic value
	value_no	int	number of characteristic value

See also:

Create Characteristics Data Table [Seite 374]

Create Internal Table [Seite 351]

Create and Fill Internal Table [Seite 352]

Save Search Result

This function is used to save characteristics or the result of a function for finding objects via classes in an internal table.

intSapFillSmlTable (order,rcvstr,rcvlen).

Input/Output	Parameter	Type	Description
Input	order	int	1 - table for finding objects via classes 2 - table of classification data of an object
	rcvstr	char*	data string from SAP System
	rcvlen	int	length of data string

See also:

Create Send String for Characteristics Data (R/2 and R/3) [Seite 366]

Create Characteristics Data Table [Seite 374]

Delete Search Result (R/2) [Seite 375]

Load Classification Data for Characteristic Number (R/2) [Seite 376]

Load Classification Data for Characteristic Number (R/3) [Seite 378]

Save Characteristic Value and Description [Seite 385]

Matchcode Functions

The following matchcode functions are supported in the CAD function library:

Count Matchcode Records [Seite 390]

Create and Fill Matchcode Table [Seite 391]

Create Send String for Matchcode [Seite 392]

Load Matchcode Fields [Seite 393]

Save Matchcode Field Value [Seite 394]

Count Matchcode Records

This function is used to check and count the records in the matchcode table.

intSapGetMcdDescription (order,string)

Input/Output	Parameter	Type	Description
Input	order	int	1 - number of defined matchcodes or number of characteristics (R/2 classification) 2 - number of selected objects
Output	string	char**	number

See also:

Create and Fill Matchcode Table [Seite 391]

Load Matchcode Fields [Seite 393]

Create Send String for Matchcode [Seite 392]

Save Matchcode Field Value [Seite 394]

Create and Fill Internal Matchcode Table

This function is used to create an internal table of the descriptions of matchcode IDs and matchcode fields and fill this table.

intSapCreateFillTableMcd (order,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - existing matchcodes in the SAP System (sub-process="ID") 2 - fields of the selected matchcode (sub-process="RF")
	rcvstr	char*	data string from SAP System
	rcvlen	long	length of data string

See also:

Save Matchcode Field Value [Seite 394]

Create Send String for Matchcode [Seite 392]

Create Send String for Matchcode

This function is used to create the send string for finding objects via matchcode. Memory for the send string is allocated dynamically.

intSapCreateSendstrMcd (order,pcode,ucode,sendstr)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table with matchcode data
	pcode	char*	Process code
	ucode	char*	Sub-process code
Output	sendstr	char**	Send string for upload to SAP

See also:

Create and Fill Matchcode Table [Seite 391]

Create Send String for Matchcode [Seite 392]

Load Matchcode Fields [Seite 393]

Count Matchcode Records [Seite 390]

Save Matchcode Field Value [Seite 394]

Load Matchcode Fields

This function reads the attribute fields of the matchcode from an internal table.

intSapGetMcdValue (attname,record_no,value)

Input/Output	Parameter	Type	Description
Input	attname	char*	name of attribute: „ID“ - for matchcode ID „SUBID“ - for matchcode SUBID „TEXT01“ - for description (list) „TEXT02“ - for description (selection) „WERT“ - for the value
	record_no	int	number of data record in table
Output	value	char**	value of specified attribute

See also:

Create and Fill Matchcode Table [Seite 391]

Count Matchcode Records [Seite 390]

Create Send String for Matchcode [Seite 392]

Save Matchcode Field Value [Seite 394]

Save Matchcode Field Value

This function is used to save the value of a matchcode attribute field to the internal matchcode table.

intSapWriteMcdValue (order,number,value)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table with matchcode data
	number	int	attribute number
	value	char*	value of specified attribute

See also:

Create and Fill Matchcode Table [Seite 391]

Create Send String for Matchcode [Seite 392]

Online Help Functions

The following online help functions are supported in the CAD toolbox library:

Count Table Entries (Help Functions) [Seite 396]

Internal Table for Online Help [Seite 397]

Create Send String for Online Help [Seite 398]

Create Send String for Long Text of SAP Message (R/3) [Seite 399]

Internal Table for Online Help [Seite 400]

Load Online Help Data [Seite 401]

Count Table Entries (Help Functions)

This function describes the internal help table and computes the number of rows or columns.

intSapGetHlpDescription (order,string)

Input/Output	Parameter	Type	Description
Input	order	int	1 - number of columns 2 - number of rows 3 - column number with selectable value (R/3 > 3.0)
Output	number	char**	number

See also:

Internal Table for Online Help [Seite 400]

Internal Table for Online Help [Seite 397]

Load Online Help Data [Seite 401]

Fill Internal Table for Online Help

This function is used to create an internal table for storing online documentation and check tables.

intSapCreateHlpTable (order,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - check table 2 - online documentation table
	rcvstr	char*	data received from SAP
	rcvlen	long	length of data received

See also:

Internal Table for Online Help [Seite 400]

Count Table Entries (Help Functions) [Seite 396]

Load Online Help Data [Seite 401]

Create Send String for Online Help

This function is used to create a string containing the data that must be sent to the SAP System when function SapHlprq(„SD“...) or SapHlprq(„SE“...) is called.

intSapCreateSendstrHelp (pcode,ucode,direction,field_name,sendstr)

Input/Output	Parameter	Type	Description
Input	pcode	char*	Process code
	ucode	char*	Sub-process code
	direction	char*	upload or download (“U” or “D”)
	field_name	char*	field name
Output	sendstr	char**	Send string for upload to SAP

See also:

Internal Table for Online Help [Seite 400]

Count Table Entries (Help Functions) [Seite 396]

Load Online Help Data [Seite 401]

Create Send String for Long Text of SAP Message (R/3)

This function is used to create a send string for loading the long text of a specific SAP message. An SAP message is returned by each SAP function in the CAD function library.

intSapCreateSendstrMessageR3 (sapmes,meslen,sendstr)

Input/Output	Parameter	Type	Description
Input	sapmes	char*	SAP message (returned by SAP function in CAD function library)
	meslen	int	length of SAP message (hd.meslen)
Output	sendstr	char**	Send string for upload to SAP

Fill Internal Table for Online Help

This function fills the existing internal help table with values from the SAP check table.

intSapFillHlpTable (order,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - check table
	rcvstr	char*	data received from SAP
	rcvlen	int	length of data received

See also:

Internal Table for Online Help [Seite 397]

Count Table Entries (Help Functions) [Seite 396]

Load Online Help Data [Seite 401]

Load Online Help Data

This function loads online documentation from the internal table. If you want to read the online documentation table, the parameter COLUMN must be 1.

intSapGetHlpData (order,column,row,hlpdata)

Input/Output	Parameter	Type	Description
Input	order	int	1 - online help data
	column	int	number of columns in internal table
	row	int	number of rows in internal table
Output	hlpdata	SAPSML*	Help Data SAPSML [Extern]

See also:

Internal Table for Online Help [Seite 400]

Internal Table for Online Help [Seite 397]

Count Table Entries (Help Functions) [Seite 396]

Mail Functions (R/3 2.1/2.2)

The toolbox mail functions only apply to an R/3 System of release 2.1 or 2.2. If you are using an R/3 system of release 3.0, you can handle the mail functions in the same way as Master Record Functions [Seite 349].

The following mail functions are supported in the CAD toolbox library:

Count Table Entries (Mail Functions) [Seite 403]

Create Send String for Mail Data [Seite 404]

Create Internal Table for Mail Data [Seite 405]

Load Mail Data from Table [Seite 406]

Save Mail Data to Table [Seite 409]

Count Table Entries (Mail Functions)

This function checks the internal mail table and counts the entries in the mail table.

intSapDescribeMailTable (order,string)

Input/Output	Parameter	Type	Description
Input	order	int	1 - mailbox (inbox, outbox, trash) 2 - mail text 3 - mail header 4 - mail parameter 1 5 - mail parameter 2 6 - mail recipient list
Output	number	char**	number

See also:

Create Internal Table for Mail Data [Seite 405]

Create Send String for Mail Data

This function is used to create a send string containing data described at the specified position in the mail table. The function allocates memory for the string dynamically.

intSapCreateSendstrMail (order, position,sendstr)

Input/Output	Parameter	Type	Description
Input	order	int	1 - mailbox (inbox, outbox) 2 - mailbox (trash)
	position	int	position in the list
Output	sendstr	char**	send string for upload

See also:

Create Internal Table for Mail Data [Seite 405]

Create Internal Table for Mail Data

This function is used to create an internal table for mail functions and saves the data received to the table.

intSapCreateMailTable (order,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - mailbox (inbox,outbox,trash) 2 - mail text 3 - mail header 4 - mail parameter 1 5 - mail parameter 2 6 - mail recipient list
	rcvstr	char*	data received from SAP
	rcvlen	long	length of data received

See also:

Load Mail Data from Table [Seite 406]

Save Mail Data to Table [Seite 409]

Load Mail Data from Table

This function reads data from the internal mail table. You can use either the field name or the index of the field you require to access the table.

intSapGetMailData (order,fldind,fldnam,number,string)

Input/Output	Parameter	Type	Description
Input	order	int	1 - mailbox (inbox,outbox,trash) 2 - mail text 3 - mail header 4 - mail parameter 1 5 - mail parameter 2 6 - mail recipient list
	field_index	int	index of requested field
	field_name	char*	Mail Field Names [Seite 407]
	position	int	position in the list
Output	maildata	char**	field value

Mail Field Names

Mailbox	FOLTP	Folder type
	FOLYR	Folder year
	FOLNO	Folder number
	DOCTP	Object type
	DOCYR	Object year
	DOCNO	Object number
	DOCDES	Object description
	SNDNAM	Sender name
	RECNAM	Forwarder name
	RECCNT	Number of recipients
	OBJRD	Object has been viewed
	OBJAC	Recipient must take action
	VMTYP	Processing type
	RESUB	Resubmitted object
	SNDEX	Send express
Mail text	CONT	
Mail header	FOLTP	Folder type
	FOLYR	Folder year
	FOLNO	Folder number
	DOCTP	Object type
	DOCYR	Object year
	DOCNO	Object number
	DOCNAM	Object name
	DOCDES	Object description
	DOCLA	Object language
	OWNNAM	Owner name
	SNDNAM	Sender name
	CRDAT	Created on (date)
	CRTIM	Created at (time)
	OBJSRT	Sort field
	OBJPRI	Priority
	OBJSNS	Sensitivity (private, functional,..)

Mail Field Names

	VMTYP	Processing type
	ACNAM	Report name,FM,DB,TCode
	OBJRD	Object has been viewed
	OBJRR	Object requires reply
	OBJAC	Recipient must take action
	RESUB	Resubmitted object
	SNDEX	Send express
Mail parameter I	PARNAM	Parameter name
	PAROPT	Parameter option
	PARLOW	Value (low)
	PARHIG	Value (high)
Mail parameter II	PARNAM	Parameter name
	PARVAL	Value
Mail recipient list	RECNAM	Recipient name
	FORNAM	Forwarder name
	OBJRD	Object has been viewed
	RDDAT	Date read
	RDTIM	Time read
	SNDPRI	Send priority
	OBJAC	Recipient must take action
	ACDAT	Date by which action must be taken
	ACTIM	Time by which action must be taken
	ACALL	To do by all recipients
	ALONE	To do by one recipient in the group
	ATDAT	Date on which action was taken
	ATTIM	Time on which action was taken

Save Mail Data to Table

This function saves data to the internal mail table. You can use either the field name or the index of the field you require to access the table.

intSapWriteMailValue (order, action, field_name,value_no,value)

Input/Output	Parameter	Type	Description
Input	order	int	1 - mailbox (inbox, outbox, trash) 2 - mail text 3 - mail header 4 - mail parameter 1 5 - mail parameter 2 6 - mail recipient list
	action	int	0 - not currently in use
	field_name	char*	Mail Field Names [Seite 407]
	value_no	int	value number
	value	char*	field value

See also:

Load Mail Data from Table [Seite 406]

User-Defined Functions

The following user-defined functions are supported in the CAD toolbox library:

Create Internal Table for User-Defined Functions [Seite 411]

Create Internal Table for User-Defined Functions

This function is used to create an internal table for a user-defined function. The names of the attributes are “PARAx”. ‘x’ is the number of the attribute. You can use this table in the same way as the standard internal tables defined with CIM,CID and CIU.

intSapCreateFillUserTable (order,no_attributs,rcvstr,rcvlen)

Input/Output	Parameter	Type	Description
Input	order	int	1 - table of user-defined data
	no_attributs	int	number of attributes in the user-defined function
	rcvstr	char*	data received from SAP
	rcvlen	long	length of data received

See also:

Load Attribute Name [Seite 346]

Load Attribute Value for Record Number [Seite 357]

Load Attribute Value for Attribute Name [Seite 358]

Change Table Value [Seite 360]

Find Attribute [Seite 348]

Save Configuration Tables [Seite 343]

SaveTable Values [Seite 361]

Appendix

Files [Seite 413]

Structures [Seite 428]

Tables [Seite 437]

Files

CADTMES.DAT [Seite 414]
CADTTAB.DAT [Seite 418]
CADTOOLC.H [Seite 420]

CADTMES.DAT

```
/*-------------------------------------------------------------------
-----------------------------------*/
/*
                                                        */
/*  SAP AG Walldorf
                                                        */
/*  SAP  Systeme, Anwendungen und Produkte in der Datenverarbeitung  */
/*
                                                        */
/*  (C) Copyright SAP AG 1995
                                                        */
/*
                                                        */
/*-------------------------------------------------------------------
-----------------------------------*/
/*  Messagefile for toolbox
                                                        */
/*-------------------------------------------------------------------
-----------------------------------*/
```

E 001 X SapCreateTable	TB001 Function code does not exist in table
E 002 K SapCreateTable	TB002 Set does not exist in table
E 003 P	TB003 Order does not exist
E 004 E SapCreateFillTable	TB004 Data cannot be saved to structure
E 005 E SapCreateFillTableMcd	TB005 List of matchcodes cannot be saved
E 006 E SapCreateFillTableMcd	TB006 Description of matchcodes cannot be saved
E 007 A SapCreateSendstr	TB007 Internal structure is not defined
E 008 P	TB008 Memory allocation failure
E 009 A SapCreateSendstrSml	TB009 Internal structure is not defined
E 010 A SapCreateSendstrMcd	TB010 Internal structure is not defined
E 011 P SapCreateSmlTable	TB011 Release of the toolbox is not compatible

E 012 E SapCreateSmlTable	TB012 Characteristic structure cannot be created
E 013 E SapCreateHlpTable	TB013 HELP structure cannot be created
E 014 A SapDeleteSmlSelection	TB014 Characteristic structure does not exist
E 015 A SapDescribeTable	TB015 Matchcode structure does not exist
E 016 A SapDescribeTable	TB016 CID structure does not exist
E 017 A SapDescribeTable	TB017 CIU structure does not exist
E 018 P SapFillInterface	TB018 Release of the toolbox is not compatible
E 019 E SapFillHlpTable	TB019 HELP structure is not created
E 020 E SapFillHlpTable	TB020 Cannot write to HELP structure
E 021 A SapFillSmlTable	TB021 Characteristic structure does not exist
E 022 E SapFillSmlTable	TB022 Cannot write to characteristic structure
E 023 A SapGetAttributNameValue	TB023 Matchcode structure does not exist
E 024 A SapGetAttributNameValue	TB024 CID structure does not exist
E 025 A SapGetAttributNameValue	TB025 CIU structure does not exist
E 026 A SapGetAttributNameValue	TB026 Record does not exist in table
E 027 A SapGetMcdValue	TB027 Matchcode structure does not exist
E 028 A SapGetMcdValue	TB028 Record does not exist in table
E 029 A SapGetMcdDescription	TB029 Matchcode structure does not exist
E 030 A SapGetSmlAttributData	TB030 Characteristic structure does not exist
E 031 A SapGetSmlAttributData	TB031 Characteristic does not exist
E 032 A SapGetHlpData	TB032 HLP structure does not exist
E 033 A SapGetHlpData	TB033 HLP does not exist
E 034 A SapGetSmlData	TB034 Characteristic structure does not exist
E 035 A SapGetSmlData	TB035 Matchcode structure does not exist

CADTMES.DAT

E 036 A SapGetSmlData	TB036 Characteristic does not exist
E 037 A SapGetHlpDescription	TB037 HLP structure does not exist
E 038 A SapGetTableValue	TB038 CIU structure does not exist
E 039 A SapGetTableValue	TB039 CID structure does not exist
E 040 A SapGetTableValue	TB040 Matchcode structure does not exist
E 041 W SapGetTableValue	TB041 Attribute does not exist in table
E 042 A SapWriteKonfiguration	TB042 Table TCIU cannot be saved
E 043 A SapWriteKonfiguration	TB043 Table TCID cannot be saved
E 044 A SapWriteKonfiguration	TB044 Table CIM cannot be saved
E 045 A SapWriteSmlValue	TB045 Characteristic structure does not exist
E 046 A SapWriteSmlValue	TB046 Matchcode structure does not exist
E 047 W SapWriteSmlValue	TB047 Characteristic does not exist
E 048 A SapWriteTableValue	TB048 CIU structure does not exist
E 049 A SapWriteTableValue	TB049 CID structure does not exist
E 050 W SapWriteTableValue	TB050 Attribute does not exist in table
E 051 A SapModifyTableValue	TB051 CIU structure does not exist
E 052 A SapModifyTableValue	TB052 CID structure does not exist
E 053 W SapModifyTableValue	TB053 Attribute does not exist in table
E 054 A SapWriteMcdValue	TB054 Matchcode structure does not exist
E 055 A SapWriteMcdValue	TB055 Matchcode does not exist
E 056 A SapGetAttributName	TB056 Table does not exist
E 057 A SapGetAttributName	TB057 Attribute does not exist
E 058 W SapSearchAttribut	TB058 Attribute does not exist in table
E 059 A SapWriteTableValue	TB059 USR structure does not exist
E 060 A SapReadTableStatic	TB060 Cannot create internal table
E 061 A SapFillTable	TB061 Internal structure is not defined
E 062 E SapFillTable	TB062 Data cannot be saved to structure
E 063 A SapCreateMailBox	TB063 Internal structure cannot be created

E 064 A SapCreateMailBox	TB064 Data cannot be saved to structure
E 065 A SapGetMailData	TB065 Mail structure does not exist
E 066 A SapGetMailData	TB066 Mail does not exist
E 067 A SapCreateSendstrMail	TB067 Internal structure is not defined
E 068 A SapGetSmlData	TB068 Mail structure does not exist
E 069 A SapDescribeMailTable	TB069 MAIL structure does not exist
E 070 A SapWriteMailValue	TB070 Mail structure does not exist
E 071 E SapWriteTableValue	TB071 Length of value exceeds definition
E 072 E SapModifyTableValue	TB072 Length of value exceeds definition
E 073 E SapFindHelpTable	TB073 Selected field does not exist in CIU/CID
E 074 E SapGetHlpDescription	TB074 Function not supported in this SAP release
E 075 E SapGetClasAttributDataR3	TB075 Entry not found in class table
E 076 E SapGetClasAttributDataR3	TB076 Class table does not exist
E 077 E SapAddSmlSel	TB077 Characteristic table does not exist
E 078 E	TB078 Error in list parameter
E 079 E SapGetSmlDataValueR3	TB079 Characteristic does not exist
E 080 E SapCreateFillClasTableR3	TB080 Class table could not be created
E 081 E SapCreateFillClasTableR3	TB081 Error filling class table
E 082 E SapCreateSenstrMessageR3	TB082 Send string could not be created
E 083 E SapCreateFillPosSmlValue	TB083 Characteristic structure could not be created
E 084 E SapFillSmlTable	TB084 Error filling object item
E 085 E SapFillSmlTable	TB085 Error filling characteristic item

CADTTAB.DAT

```
/*-------------------------------------------------------------------
-----------------------------------*/
/*
                                                        */
/*  SAP AG Walldorf
                                                        */
/*  SAP  Systeme, Anwendungen und Produkte in der Datenverarbeitung  */
/*
                                                        */
/*  (C) Copyright SAP AG 1995
                                                        */
/*
                                                        */
/*-------------------------------------------------------------------
-----------------------------------*/
D R/2 SLCLS RH SLCLSRH STNUM       003
D R/2 SLCLS RH SLCLSRH VKLAS       009
D R/2 SLCLS RH SLCLSRH KLNUM       009
D R/2 SLCLS RH SLCLSRH KLDAT       004
D R/2 SLCLS RH SLCLSRH KTX01       070
D R/2 SLCLS RH SLCLSRH KLART       001
D R/2 SLCLS RL SLCLSRL KLNUM       009
D R/2 SLCLS RL SLCLSRL KLDAT       004
D R/2 SLCLS RL SLCLSRL KTX01       070
D R/2 SLCLS RL SLCLSRL KLART       001
D R/2 SLCLS HS SLCLSHS KLNUM       009
D R/2 SLCLS HS SLCLSHS KLTYP       001
D R/2 SLCLS HS SLCLSHS KTX01       070
D R/2 SLCLS HM SLCLSHM KLNUM       009
D R/2 SLCLS HM SLCLSHM KLTYP       001
D R/2 SLCLS HM SLCLSHM KLDAT       004
D R/2 SLCLS HM SLCLSRL KTX01       070
D R/2 SLCLS HM SLCLSRL KLART       001
D R/2 SLCLS GM SLCLSGM KLNUM       009
D R/2 SLCLS GM SLCLSGM KLTYP       001
```

CADTTAB.DAT

```
D R/2 SLCLS GM SLCLSGM KLDAT        004
D R/2 SLCLS GM SLCLSGM KTX01        070
```

CADTOOLC.H

```
/*-------------------------------------------------------------------
-----------------------------------*/
/*
                                                        */
/*  SAP AG Walldorf
                                                        */
/*  SAP  Systeme, Anwendungen und Produkte in der Datenverarbeitung  */
/*
                                                        */
/*  (C) Copyright SAP AG 1995
                                                        */
/*
                                                        */
/*-------------------------------------------------------------------
-----------------------------------*/
/*  File CADTOOLC.H                                              */
/*-------------------------------------------------------------------
-----------------------------------*/
/*  header file for CAD dialog interface toolbox
                                                          */
/*-------------------------------------------------------------------
-----------------------------------*/
/*  version: 3.0C                                          */
/*-------------------------------------------------------------------
-----------------------------------*/

#ifndef _SAP_CADTOOLC_INCLUDED
#define _SAP_CADTOOLC_INCLUDED

/******************* Defines    ************************************/
```

#define	FUNCTION_SIZE	36
#define	MESTYP_SIZE	2
#define	MESSAGE_SIZE	120
#define	SETNAM_SIZE	8
#define	ATTNAM_SIZE	11

```
#define  ATTLEN_SIZE      3
#define  ATTDOKU_SIZE     40
#define  PCODE_SIZE       5
#define  UCODE_SIZE       2
#define  VERSION_SIZE     3
#define  SML_SIZE         40
#define  SML_VAL_SIZE     40
```

```
/******************** Structures ***************************************/

typedef struct {
        char    *id;
        char    *objtab;
        char    *objkey1;
        char    *objkey2;
        char    *objkey3;
        char    *objkey4;
        char    *objkey5;
        char    *objkey6;
        char    *objkey7;
        char    *objkey8;
        char    *objkey9;
        char    *objkey10;
        char    *objtext;
        char    *klasstatus;
} SAPOBJ_R3;

typedef struct {
        char    *name;
        char    *feldname;
        char    *din;
        char    *eingabe;
        char    *einheit;
        char    *format;
```

CADTOOLC.H

char	*schablone;
char	*selekt;
char	*param;
int	laenge;
int	istlen;
int	dezimalstellen;
char	*value;

} SAPSML;

typedef struct {

char	*name;
char	*bezeichnung;
char	*knz_obligat;
char	*knz_ererbt;
char	*knz_bewertet;
char	*datentyp;
char	*knz_einzelwert;
char	*knz_intervall;
char	*knz_zusaetzl_werte;
int	anzahl_stellen;
int	dezimalstellen;
char	*knz_vorzeichen;
char	*exponent_bild;
char	*exponent_darst;
char	*schablone;
char	*knz_kleinschr;
char	*ext_masseinh;
char	*dinkenbuchst;
char	*dokumentart;
char	*dokumentnummer;
char	*teildokument;
char	*dokumentversion;
char	*knz_sicht0;
char	*knz_sicht1;

char	*knz_sicht2;
char	*knz_sicht3;
char	*knz_sicht4;
char	*knz_sicht5;
char	*knz_sicht6;
char	*knz_sicht7;
char	*knz_sicht8;
char	*knz_sicht9;
char	*kriterium;
char	*suchbed;
char	*value;

} SAPSML_R3;

typedef struct

{

char	*art;
char	*name;
char	*bezeichnung;
char	*statustext;
char	*knz_standard;

} SAPCLAS_R3;

typedef struct {

char	*art;
char	*name;
char	*schlagwort;
char	*status;
char	*datum_von;
char	*datum_bis;
char	*knz_gleiche_kls;
char	*anz_zuordnungen;
char	*knz_dinkls;
char	*norm_name;
char	*norm_nummer;

CADTOOLC.H

char	*norm_bez;
char	*norm_titel1;
char	*norm_titel2;
char	*ausg_datum;
char	*vers_datum;
char	*vers_nummer;
char	*sml_leiste;
char	*dokar;
char	*doknr;
char	*doktl;
char	*dokvr;
char	
char	

} SAPCLAS_S_R3;

typedef struct {

char	*stufe;
char	*knz_unter_klas;
char	*vorg_klas;
char	*nach_klas;
char	*sw_vorg_klas;
char	*sw_nach_klas;

} SAPCLAS_H_R3;

typedef struct {

int	number;
char	mestyp[MESTYP_SIZE+1];
char	Function[FUNCTION_SIZE+1];
char	message[MESSAGE_SIZE+1];

} SAPMES;

typedef struct {

char	setname[SETNAM_SIZE+1];
char	attname[ATTNAM_SIZE+1];
char	attlen[ATTLEN_SIZE+1];
char	attdoku[ATTDOKU_SIZE+1];

```
} SAPATT;

/************** Function prototypes **********************************/

int SapCreateTable(int,char*,char*,char*,char*);

int SapCreateFillClasHierarchTableR3(int, char*, long, int*);

int SapCreateFillClasTableR3(int, char*, long, int*);

int SapCreateFillPosSmlValueTableR3(int, char*, long);

int SapCreateFillTable(int,char*,char*,char*,char*,char*,int);

int SapCreateFillTableMcd(int,char*,int);

int SapCreateSendstr(char**);

int SapCreateSendstrBomHead(int, char**);

int
SapCreateSendstrClasR3(char*,char*,char*,char*,char*,char*,char*,char*,
char**);

int SapCreateSendstrClasScopeR3(char*, char*,
char*,char*,char*,char*,char**);

int
SapCreateSendstrClasSearchR3(int,char*,char*,char*,char*,char*,char*,ch
ar*,

                                  char**);

int
SapCreateSendstrClasSelectR3(char*,char*,char*,char*,char*,char*,char*,
char*,

                                 char*, char*,char*,char**);

int SapCreateSendstrHelp(char*,char*,char*,char*,char**);

int SapCreateSendstrPosSmlValuesR3(int, char**);

int SapCreateSendstrSml(int,char**);

int SapCreateSendstrMcd(int,char*,char*,char**);

int SapCreateSendstrMessageR3(char*, int, char**);

int
SapCreateSendstrScopeClasR3(char*,char*,char*,char*,char*,char*,char*,c
har*,

                                                          char**);
```

CADTOOLC.H

```
int SapCreateSmlTable(int,char*,int);
int SapCreateSmlSelectTable(char*);
int SapCreateHlpTable(int,char*,int);
int SapDeleteSmlSelection();
int SapDescribeInterface(int,char**);
int SapDescribeTable(int,char**);
int SapFindHelpTable( char*, char*, char*, char*, char*, char*, char*);
int SapFillHlpTable(int,char*,int);
int SapFillInterface(char*,int);
int SapFillSmlTable(int,char*,int);
int SapGetAttributNameValue(int,int,int,char*,char**);
int SapGetClasDataR3(int, int, SAPCLAS_R3*);
int SapGetClasHierarchDataR3(int, int, SAPCLAS_H_R3*);
int SapGetClasSearchDataR3(int, int, SAPCLAS_S_R3*);
int SapGetMcdDescription(int,char**);
int SapGetMcdValue(char*,int,char**);
int SapGetObjectDataR3(int, int, SAPOBJ_R3*);
int SapGetSmlAttributData(int,char*,int,SAPSML*);
int SapGetSmlAttributDataR3(int,char*,int,SAPSML_R3*);
int SapGetHlpData(int,int,int,SAPSML*);
int SapGetSmlData(int,int,int,SAPSML*);
int SapGetSmlDataR3(int,int,int,SAPSML_R3*);
int SapGetSmlDataValueR3(int,char*,int,int,int,char**);
int SapGetHlpDescription(int,char**);
int SapGetSmlDescription(int,char**);
int SapGetSmlValueDescriptionR3(int,int,char*,int*);
int SapGetTableValue (int,char*,int,char**);
int SapInitToolbox(int);
int SapPrintTable(int,char*);
int SapWriteKonfiguration(int,char*,int,char*);
int SapWriteSmlValue(int,int,char*,int,char*,int);
int SapWriteTableValue(int,char*,char*,int);
int SapModifyTableValue(int,char*,char*,int);
int SapGetErrorMessage(int,SAPMES*);
int SapGetAttributName(int,char*,char*,char*,char*,int,SAPATT*);
```

```
int SapSearchAttribut(int,char*,char*,char*,char*,char*,SAPATT*);
int SapWriteMcdValue(int,int,char*);
int SapCreateFillUserTable(int,int,char*,int);
int SapToolboxVersion(char**);
int SapGetMailData(int,int,char*,int,char**);
int SapCreateSendstrMail(int,int,char**);
int SapCreateMailTable(int,char*,int);
int SapWriteMailValue(int,int,char*,int,char*);
int SapDescribeMailTable(int,char**);
int SapFillTable(int,char*,int);
int SapDeleteTableEntry(int, int);

#endif /* _SAP_CADTOOLC_INCLUDED */
```

Structures

SAPOBJ_R3 [Seite 429]
SAPSML [Extern]
SAPSML_R3 [Seite 430]
SAPCLAS_R3 [Seite 432]
SAPCLAS_S_R3 [Seite 433]
SAPCLAS_H_R3 [Seite 434]
SAPMES [Seite 435]
SAPATT [Seite 436]

SAPOBJ_R3

```
typedef struct {
        char     *id;
        char     *objtab;
        char     *objkey1;
        char     *objkey2;
        char     *objkey3;
        char     *objkey4;
        char     *objkey5;
        char     *objkey6;
        char     *objkey7;
        char     *objkey8;
        char     *objkey9;
        char     *objkey10;
        char     *objtext;
        char    *klasstatus;
} SAPOBJ_R3;
```

SAPSML_R3

typedef struct {

char	*name;
char	*bezeichnung;
char	*knz_obligat;
char	*knz_ererbt;
char	*knz_bewertet;
char	*datentyp;
char	*knz_einzelwert;
char	*knz_intervall;
char	*knz_zusaetzl_werte;
int	anzahl_stellen;
int	dezimalstellen;
char	*knz_vorzeichen;
char	*exponent_bild;
char	*exponent_darst;
char	*schablone;
char	*knz_kleinschr;
char	*ext_masseinh;
char	*dinkenbuchst;
char	*dokumentart;
char	*dokumentnummer;
char	*teildokument;
char	*dokumentversion;
char	*knz_sicht0;
char	*knz_sicht1;
char	*knz_sicht2;
char	*knz_sicht3;
char	*knz_sicht4;
char	*knz_sicht5;
char	*knz_sicht6;
char	*knz_sicht7;
char	*knz_sicht8;

char	*knz_sicht9;
char	*kriterium;
char	*suchbed;
char	*value;

} SAPSML_R3;

SAPCLAS_R3

typedef struct

{

char	*art;
char	*name;
char	*bezeichnung;
char	*statustext;
char	*knz_standard;

} SAPCLAS_R3;

SAPCLAS_S_R3

typedef struct {

char	*art;
char	*name;
char	*schlagwort;
char	*status;
char	*datum_von;
char	*datum_bis;
char	*knz_gleiche_kls;
char	*anz_zuordnungen;
char	*knz_dinkls;
char	*norm_name;
char	*norm_nummer;
char	*norm_bez;
char	*norm_titel1;
char	*norm_titel2;
char	*ausg_datum;
char	*vers_datum;
char	*vers_nummer;
char	*sml_leiste;
char	*dokar;
char	*doknr;
char	*doktl;
char	*dokvr;
char	
char	

} SAPCLAS_S_R3;

SAPCLAS_H_R3

typedef struct {

char	*stufe;
char	*knz_unter_klas;
char	*vorg_klas;
char	*nach_klas;
char	*sw_vorg_klas;
char	*sw_nach_klas;

} SAPCLAS_H_R3;

SAPMES

```
typedef struct {
    int number;
    char mestyp[MESTYP_SIZE+1];
    char function[FUNCTION_SIZE+1];
    char message[MESSAGE_SIZE+1];
} SAPMES;
```

SAPATT

```
typedef struct {
    char    setname[SETNAM_SIZE+1];
    char    attname[ATTNAM_SIZE+1];
    char    attlen[ATTLEN_SIZE+1];
    char    attdoku[ATTDOKU_SIZE+1];
} SAPATT;
```

Tables

TCIM

TCIU [Seite 447]
TCIM [Seite 438]
TCIT [Seite 440]
TCID [Seite 444]
TCIU [Seite 447]

TCIM

PrcsCode	SubP	Dir.	Set name	Description
BOMCN	MK	D	BOCNOLD	Display old BOM header data
BOMCN	MK	U	BOCNNEW	Add changed BOM header data
BOMCN	MP	U	BOCNDAT	Change BOM with item numbers
BOMCN	SP	D	ITEMDAT	Display old BOM item data
BOMCN	SP	U	ITEMDAT	Add changed item data
BOMCR	MP	U	BOCRDAT	Create new BOM with item numbers
BOMCR	SP	U	ITEMDAT	Add BOM item data
BOMRQ	LB	D	BORQDAT	Display BOM header data
BOMRQ	LB	U	BORQDAT	Display BOM level by level
BOMRQ	LP	D	ITEMLIS	Display BOM item data
BOMRQ	MB	D	BORQDAT	Display BOM header data
BOMRQ	MB	U	BORQDAT	Display multi-level BOM
BOMRQ	MP	D	ITEMLIS	Display BOM item data
BOMRQ	MR	D	BORQDAT	Display BOM header data
BOMRQ	MR	U	BORQDAT	Display single-level BOM
BOMRQ	QB	D	BORQDAT	Display BOM header data
BOMRQ	QB	U	BORQDAT	Display summarized BOM
BOMRQ	QP	D	ITEMLIS	Display BOM item data
BOMRQ	SP	D	ITEMLIS	Display BOM item data
BOMWU	RL	D	WH_USED	Display where-used list
BOMWU	RL	U	WH_USED	Display where-used list
CHGCH	MD	D	CHG_DAT	Display old change master data
CHGCH	MD	U	CHGDAT1	Add changed change master data
CHGCH	MR	U	CHG_KEY	Change change master without class
CHGCR	MR	D	CHG_KEY	Display function message
CHGCR	MR	U	CHG_DAT	Create change master
CHGRQ	MR	D	CHG_DAT	Display change master data
CHGRQ	MR	U	CHG_KEY	Display change master without class
DOCCH	MD	D	DCRQDAT	Display old document info record data
DOCCH	MD	U	DCCRDAT	Create change document info record
DOCCH	MR	U	DCRQDAT	Change document record without class
DOCCR	MR	D	DOC_KEY	Display function message
DOCCR	MR	U	DCCRDAT	Create document info record
DOCRQ	DL	D	DCRQLIS	Display selected document info rec.
DOCRQ	DL	U	DSEARCH	Find document via retrieval function
DOCRQ	MC	D	DCRQDAT	Display document record data
DOCRQ	MC	U	DCRQDAT	Display document record with class
DOCRQ	MR	D	DCRQDAT	Display document record data
DOCRQ	MR	U	DCRQDAT	Display document record without class
DOCRQ	OS	D	DCRQLIS	Display selected document info records
DOCRQ	OS	U	DSEARCH	Find document with allocated objects
DOCRQ	RL	D	DCRQLIS	Find document record via matchcode
EQICH	MC	U	E_CHG_C	Change equipment with classification
EQICH	MD	D	EQRQDAT	Display old equipment master data
EQICH	MD	U	EQMDDAT	Add changed equipment master data

EQICH	MR	U	EQRQDAT	Change equipment without classification
EQICR	MR	D	EQI_KEY	Display function message
EQICR	MR	U	EQCRDAT	Create equipment
EQIRQ	MC	D	EQRQDAT	Display equipment
EQIRQ	MC	U	EQRQDAT	Display equipment with classification
EQIRQ	MR	D	EQRQDAT	Display equipment
EQIRQ	MR	U	EQRQDAT	Display equipment without classification
EQIRQ	RL	D	EQI_MCD	Find equipment via matchcode
MATCH	MC	U	M_CHG_C	Change material master with class.
MATCH	MD	D	MTRQDAT	Display Old Material Master Data
MATCH	MD	U	MTCRDAT	Create changed material master data
MATCH	MR	U	MTRQDAT	Change material master without class.
MATCR	MA	U	MTCRDAT	Create material master after reservation
MATCR	MR	D	MAT_KEY	Display function message
MATCR	MR	U	MTCRDAT	Create material master
MATRQ	MC	D	MTRQDAT	Display material master data
MATRQ	MC	U	MTRQDAT	Display material master record with class.
MATRQ	MR	D	MTRQDAT	Display material master data
MATRQ	MR	U	MTRQDAT	Display material record without class.
MATRQ	RL	D	MTRQLIS	Find material via matchcode
PMNCR	MR	U	PMCRDAT	Create maintenance notification
SLCLS	HS	D	CLSSDAT	Find classes without a superior class
SMAIL	IB	D	FOLDATA	Display documents in inbox
SMAIL	MC	D	ATTHEAD	Display asset header data
SMAIL	MH	D	DOCHEAD	Display document header data
SMAIL	OB	D	FOLDATA	Read mail outbox
SMAIL	RC	U	DOCRECL	Send document (recipient)
SMAIL	RL	D	DOCRECL	Display recipient list for document
SMAIL	SM	U	DOCHEAD	Send document (header data)
SMAIL	WB	D	FOLDATA	View trash in mail
TPCCH	MC	U	F_CHG_C	Change functional location with class.
TPCCH	MD	D	FLRQDAT	Display old functional location data
TPCCH	MD	U	FLMDDAT	Add changed functional location data
TPCCH	MR	U	FLRQDAT	Change functional location without class.
TPCCR	MR	D	FLC_KEY	Display function message
TPCCR	MR	U	FLCRDAT	Create functional location
TPCRQ	MC	D	FLRQDAT	Display functional location data
TPCRQ	MC	U	FLRQDAT	Display functional location with class.
TPCRQ	MR	D	FLRQDAT	Display functional location data
TPCRQ	MR	U	FLRQDAT	Display functional location without class.
TPCRQ	RL	D	FLC_MCD	Find functional location via matchcode

TCIT

Field name	Screen No.	ScrFieldName	Len
ALTERNATIV	SAPMC29L0100	RC29L-STLAL	002
APPLICAT	SAPMC29L0100	RC29L-CAPID	004
ASSEMBLY	SAPMC29L0105	RC29L-BAGRP	018
ASSET	SAPMILA07000	ILOA-ANLNR	012
ASSIGNED	SAPLCTMS0160	RCTMS-MEMIT	001
BASE_UNIT	SAPMM03M0200	MARA-MEINS	003
BASIC_MAT	SAPMM03M0200	MARA-WRKST	014
BOM_QUAN	SAPLCSDI0110	RC29K-BMENG	017
BOM_STATUS	SAPLCSDI0110	BOM_STATUS	002
BOM_USAGE	SAPLCSDI0100	RC29N-STLAN	001
BOM_USAGE	SAPMC29L0205	RC29L-STLAN	001
CARRIER	SAPLCV000102	DRAW-DTTRG	010
CAUSE_CODE	SAPMIWO02100	VIQMUR-URCOD	004
CAUSE_GRP	SAPMIWO02100	VIQMUR-URGRP	008
CHANGE_NO	SAPLCSDI0100	RC29N-AENNR	012
CHANGE_NO	SAPLCV000101	DRAW-AENNR	012
CHANGE_NO	SAPLCVIN0400	DRAW-AENNR	012
CHANGE_NO	SAPMC29C0100	RC29A-AENNR	012
CHANGE_NO	SAPMC29L0100	RC29L-AENNR	012
CHG_REASON	SAPMC29C0110	RC29A-AEGRU	040
CHG_STATUS	SAPMC29C0110	RC29A-AENST	002
CLASS_NO	SAPLCLFM0501	RMCLF-CLASS	018
CLASS_TYPE	SAPLCLFM0501	RMCLF-KLART	003
CONFIDENT	SAPLSO010200	SOS01-S_OBJPR	001
CONST_MON	SAPMIEQ00102	EQUI-BAUMM	002
CONST_YEAR	SAPMIEQ00102	EQUI-BAUJJ	004
COST_CENT	SAPMILA07000	ILOA-KOSTL	010
DEPARTMENT	SAPLCVIN0400	DRAW-LABOR	003
DESCRIPT_A	SAPLCSDI0110	RC29K-STKTX	040
DESCRIPT_B	SAPLCSDI0110	RC29K-ZTEXT	040
DESCRIPT_C	SAPMC29C0110	RC29A-AETXT	040
DESCRIPT_D	SAPLCV000101	DRAT-DKTXT	040
DESCRIPT_E	SAPMIEQ00110	EQKT-EQKTX	040
DESCRIPT_M	SAPMM03K0100	MAKT-MAKTX	040
DESCRIPT_M	SAPMM03K0110	MAKT-MAKTX	040
DESCRIPT_M	SAPMM03M0200	MAKT-MAKTX	040
DESCRIPT_N	SAPMIWO01010	VIQMEL-QMTXT	040
DESCRIPT_T	SAPMILO02120	IFLO-PLTXT	040
DESCR_MEAS	SAPMIWO04000	VIQMSM-MATXT	040
DESCR_POS	SAPMIWO03000	VIQMFE-FETXT	040
DIN_FORMAT	SAPMM03M0200	MARA-FORMT	004
DIRECT	SAPMC29L0200	RC29L-DIRKT	001
DISP_APPL	SAPLCV000101	DRAW-DAPPL	003
DOC_NAME	SAPLSO010200	SOS01-S_OBJNAM	012
DOC_NUMBER	SAPLCV000100	DRAW-DOKNR	025

DOC_NUMBER	SAPLCVIN0400	DRAW-DOKNR	025
DOC_OFFICE	SAPLCV000101	DRAW-LABOR	003
DOC_PART	SAPLCV000100	DRAW-DOKTL	003
DOC_PART	SAPLCVIN0400	DRAW-DOKTL	003
DOC_PATH	SAPLCV000102	DRAW-FILEP	070
DOC_STATUS	SAPLCV000101	TDWST-STABK	002
DOC_STATUS	SAPLCVIN0400	TDWST-STABK	002
DOC_TITLE	SAPLSO010200	SOS01-S_OBJDES	050
DOC_TYPE	SAPLCV000100	DRAW-DOKAR	003
DOC_TYPE	SAPLCVIN0400	DRAW-DOKAR	003
DOC_TYPE	SAPLSO010200	SOS01-S_OBJTP	003
DOC_VERS	SAPLCV000100	DRAW-DOKVR	002
DOC_VERS	SAPLCVIN0400	DRAW-DOKVR	002
EQUIPMENT	SAPLCV000204	EQUI-EQUNR	018
EQUIPMENT	SAPMIEQ00100	RM63E-EQUNR	018
EQUIPMENT	SAPMIWO01010	RIWO1-EQUNR	018
EQUI_BOM	SAPMC29L0200	RC29L-EQUTP	001
EQUI_CATEG	SAPMIEQ00100	RM63E-EQTYP	001
EQUI_INST	SAPMILO07030	RILO0-IEQUI	001
EQUI_DATE	SAPMIEQ00100	RM63E-DATSL	010
EXP_DATE	SAPLSO010200	SOS01-S_DLDAT	010
FLC_CATEG	SAPMILO01110	IFLO-FLTYP	001
FLC_COPY	SAPMILO01110	RILO0-TPLCP	030
FLC_REF	SAPMILO01110	IRLO-TRPNR	030
FLC_ROOMNR	SAPMILA07000	ILOA-MSGRP	008
FLC_SUPER	SAPMILO01110	RILO0-TPLMA_DFLT	030
FUNCTIONAL	SAPLSO010200	SOS01-S_OBJFC	001
FUNC_LOC	SAPLCV000205	IFLOT-TPLNR	030
FUNC_LOC	SAPLIEL10100	RM63A-TPLNEW	030
FUNC_LOC	SAPMILO01110	IFLO-TPLNR	030
FUNC_LOC	SAPMIWO01010	RIWO1-TPLNR	030
GROSS_WGHT	SAPMM03M0200	MARA-BRGEW	017
HIGH_EQUI	SAPMIEQ00106	EQUZ-HEQUI	018
INDUSTRY	SAPMM03M0060	RM03M-MBRSH	001
IND_BLIND	SAPLSO040500	SOS04-L_SGK	001
IND_COPY	SAPLSO040100	SOS04-L_SKP	001
IND_EXPRES	SAPLSO040100	SOS04-L_SEX	001
IND_NOPRI	SAPLSO040500	SOS04-L_SKD	001
IND_NO_FWD	SAPLSO040500	SOS04-L_SKW	001
IND_OUTBOX	SAPLSO040100	SOS04-S_OUTFL	001
IND_ REPLY	SAPLSO040500	SOS04-L_SAE	001
INHERITED	SAPLCTMS0160	RCTMS-MEMMV	001
ITEM_CATEG	SAPLCSDI0140	RC29P-POSTP	001
ITEM_CATEG	SAPMC29L0105	RC29L-POSTP	001
ITEM_CATEG	SAPMC29L0205	RC29L-POSTP	001
ITEM_NO	SAPLCSDI0140	RC29P-POSNR	004
I_D_DATE	SAPLIEL10100	RM63A-DATUM	010
LOCATION	SAPMILA07000	ILOA-STORT	010
LOC_PLANT	SAPMILA07000	ILOA-SWERK	004
LVL_CLASS	SAPMM03M0070	BMM00-XEIC1	001

TCIT

LVL_COST	SAPMM03M0070	BMM00-XEIG1	001
LVL_COST	SAPMM03M0070	BMM00-XEIL1	001
LVL_ENGIN	SAPMM03M0070	BMM00-XEIK1	001
LVL_MRP_1	SAPMM03M0070	BMM00-XEID1	001
LVL_MRP_2	SAPMM03M0070	BMM00-XEID2	001
LVL_MRP_3	SAPMM03M0070	BMM00-XEID3	001
LVL_STORE	SAPMM03M0070	BMM00-XEIL1	001
MAINTPLANT	SAPMIWO02100	VIQMEL-IWERK	004
MAINT_PG	SAPMIWO02100	VIQMEL-INGRP	003
MANUFACTOR	SAPMIEQ00102	EQUI-HERST	030
MANUFACT_C	SAPMIEQ00102	EQUI-HERLD	003
MARK_DISM	SAPLIEL10100	RM63A-AUSKZ	001
MARK_INST	SAPLIEL10100	RM63A-EINKZ	001
MATERIAL	SAPLCSDI0100	RC29N-MATNR	018
MATERIAL	SAPLCSDI0140	RC29P-IDNRK	018
MATERIAL	SAPLCV000201	MARA-MATNR	018
MATERIAL	SAPMC29L0100	RC29L-MATNR	018
MATERIAL	SAPMC29L0200	RC29L-MATNR	018
MATERIAL	SAPMM03M0060	RM03M-MATNR	018
MAT_BOM	SAPMC29L0200	RC29L-MATTP	001
MAT_TYPE	SAPMM03M0060	RM03M-MTART	004
MM/PP_STAT	SAPMM03M0249	MARC-MMSTA	002
MRP_TYPE	SAPMM03M0248	MARC-DISMM	002
NET_WEIGHT	SAPMM03M0200	MARA-NTGEW	017
NOTIF_TYPE	SAPMIWO00100	RIWO00-QMART	002
NO_CHANGE	SAPLSO010200	SOS01-S_CHA_NOT	001
OPTIONAL	SAPLCTMS0160	RCTMS-MEOPT	001
PLANT	SAPLCSDI0100	RC29N-WERKS	004
PLANT	SAPMC29L0100	RC29L-WERKS	004
PLANT	SAPMC29L0205	RC29L-WERKS	004
PLANT	SAPMM03M0080	RM03M-WERKS	004
PRIORITY	SAPLSO010200	SOS01-S_OBJPRI	001
PRIORITY	SAPMIWO02100	VIQMEL-PRIOK	001
PROTOCOL	SAPLCVST0203	DRAP-PROTF	020
P_DOC_NO	SAPLCV000101	DRAW-PRENR	025
P_DOC_PART	SAPLCV000101	DRAW-PRETL	003
P_DOC_TYPE	SAPLCV000101	DRAW-PREAR	003
P_DOC_VERS	SAPLCV000101	DRAW-PREVR	002
QUANTITY	SAPLCSDI0140	RC29P-MENGE	018
RECIPIENT	SAPLSO040100	SOS04-L_ADR_NAME	241
REC_TYPE	SAPLSO040100	SOS04-L_ART	001
REQUIRED	SAPLCTMS0160	RCTMS-MEMUS	001
REQ_QUANT	SAPMC29L0100	RC29L-EMENG	017
REVISION	SAPMC29L0100	RC29L-REVLV	002
SERIAL_NUM	SAPMIEQ00102	EQUI-SERGE	030
SINGL_INST	SAPMILO07030	RILO0-EINZL	032
SIZE/DIMEN	SAPMM03M0200	MARA-GROES	032
SORT_FIELD	SAPLSO010200	SOS01-S_OBJSRT	010
SORT_IND	SAPLCSDI0140	RC29P-SORTF	010
SPARE_PART	SAPLCSDI0131	RC29P-ERSKZ	001

STANDARD	SAPLSO010200	SOS01-S_OBJCO	001
STARTDAT	SAPMIWO02100	VIQMEL-AUSVN	010
STARTTIME	SAPMIWO02100	VIQMEL-AUZTV	008
STD_CLASS	SAPLCLFM0501	RMCLF-STDCL	001
STORE_LOC	SAPMM03M0080	RM03M-LGORT	004
STRUCT_IND	SAPMILO01110	RILO0-TPLKZ	005
TECH_IDENT	SAPMIEQ00106	EQUZ-TIDNR	025
UNASSIGNED	SAPLCTMS0160	RCTMS-MEOHN	001
UNIT_WGHT	SAPMM03M0200	MARA-GEWEI	003
USER	SAPLCV000101	DRAW-DWNAM	012
USER	SAPLCVIN0400	DRAW-DWNAM	012
VALID_FROM	SAPMC29C0110	RC29A-DATUV	010
VALID_FROM	SAPMC29L0100	RC29L-DATUV	010
VALID_FROM	SAPMC29L0200	RC29L-DATUV	010
VALID_TO	SAPMC29L0200	RC29L-DATUB	010
VIA_CLASS	SAPMC29L0200	RC29L-UEBKL	001

TCID

Set name	Field name	Len	Field name
ATTHEAD ATTACH_ID	SOATTINFI1-ATTACH_ID	063	ID of an attachment
ATTHEAD ATTACH_TYP	SOATTINFI1-ATT_NAME	012	Document name
BOCNOLD DESCRIPT_B	RC29K-STKTX	040	Alternative BOM text
BOCNOLD DESCRIPT_A	RC29K-ZTEXT	040	BOM text
BOCNOLD BOM_STATUS	BOM_STATUS	002	BOM status
BOCNOLD VALID_FROM	RC29K-DATUV	010	Valid-from date
BOCNOLD VALID_TO	RC29K-DATUB	010	Valid-to date
BOCNOLD DESCRIPT_M	MAKT-MAKTX	040	Material description
BORQDAT VALID_FROM	CSTMAT-DATUV	010	Valid-from/to date
BORQDAT VALID_TO	CSTMAT-DATUB	010	Valid-to date
BORQDAT BOM_STATUS	CSTMAT-STLST	002	BOM status
BORQDAT DESCRIPT_B	CSTMAT-STKTX	040	Alternative BOM text
CHG_DAT DESCRIPT_C	AENR-AETXT	040	Change number descr.
CHG_DAT VALID_FROM	AENR-DATUV	010	Valid-from date
CHG_DAT CHG_REASON	AENR-AEGRU	040	Reason for change
CHG_KEY CHANGE_NO	AENR-AENNR	012	Change number
CLSSDAT CLASS	CLCLASS-CLASS	018	Class number
CLSSDAT CLASS_TYPE	CLCLASS-KLART	003	Class type
CLSSDAT STATUS	CLCLASS-STATU	001	Class status
CLSSDAT DESCRIPT	CLCLASS-KSCHL	040	Catchwords
DCRQDAT DESCRIPT_D	DRAT-DKTXT	040	Description of document
DCRQDAT DATA_CARR	DRAW -DTTRG	010	Name of data carrier
DCRQDAT DISP_APPL	DRAW-DAPPL	003	Application
DCRQDAT PROTOCOL	DRAW-DOKVR	002	Document version
DCRQDAT USER	DRAW-DWNAM	012	Name of user responsible
DCRQDAT DOC_STATUS	DRAW-DOKST	002	Document status
DCRQDAT ORIGINAL	DRAW-FILEP	070	Original of document
DCRQLIS DOC_NUMBER	DRAW-DOKNR	025	Document number
DCRQLIS DOC_TYPE	DRAW-DOKAR	003	Document type
DCRQLIS DOC_PART	DRAW-DOKTL	003	Document part
DCRQLIS DOC_VERS	DRAW-DOKVR	002	Document version
DCRQLIS DESCRIPT_D	DRAT-DKTXT	040	Description of document
DCRQLIS DOC_STATUS	DRAW-DOKST	002	Document status
DOCHEAD SENDER	SOFOLENTI1-CREAT_NAME	012	Created by
DOCHEAD READ_FLAG	SOFOLENTI1-READ	001	Obj. already viewed
DOCHEAD CRE_DATE	SOFOLENTI1-CREAT_DATE	010	SAPoffice: date created
DOCRECL RECIPIENT	SORECLSTI1-RECEIVER	999	SAPoffice: recipient name
DOCRECL REC_TYPE	SORECLSTI1-REC_TYPE	001	Indicator for a recipient
DOCRECL FORWARDER	SORECLSTI1-FORW_NAM	012	Forwarder: name
DOCRECL IND_EXPRES	SORECLSTI1-EXPRESS	001	Send express
DOCRECL IND_COPY	SORECLSTI1-COPY	001	Send as a copy
DOCRECL IND_BLIND	SORECLSTI1-BLIND_COPY	001	Send as a blind copy
DOCRECL IND_READ	SORECLSTI1-READ	001	Object already viewed

DOCRECL READ_DATE	SORECLSTI1-READ_DATE	010	Date document viewed
DOCRECL READ_TIME	SORECLSTI1-READ_TIME	008	Time document viewed
DOC_KEY DOC_NUMBER	DRAW-DOKNR	025	Document number
DOC_KEY DOC_TYPE	DRAW-DOKAR	003	Document type
DOC_KEY DOC_PART	DRAW-DOKTL	003	Document part
DOC_KEY DOC_VERS	DRAW-DOKVR	002	Document version
EQI_KEY EQUIPMENT	EQUI-EQUNR	018	Equipment number
EQI_MCD EQUIPMENT	EQUI-EQUNR	018	Equipment number
EQI_MCD DESCRIPT_E	EQKT-EQKTX	040	Equipment description
EQRQDAT DESCRIPT_E	EQKT-EQKTX	040	Equipment description
EQRQDAT EQUI_TYPE	EQUI-EQTYP	001	Equipment category
EQRQDAT EQUI_CATEG	EQUI-EQART	010	Equipment type
EQRQDAT CONST_MON	EQUI-BAUMM	002	Month of construction
EQRQDAT CONST_YEAR	EQUI-BAUJJ	004	Year of construction
EQRQDAT MANUFACTOR	EQUI-HERST	030	Equipment manufacturer
EQRQDAT MANUFACT_C	EQUI-HERLD	003	Country of manufacture
EQRQDAT HIGH_EQUI	EQUZ-HEQUI	018	Superior equipment
EQRQDAT SERIAL_NUM	EQUI-SERNR	018	Serial number
EQRQDAT TECH_IDENT	EQUZ-TIDNR	025	Technical identification
FLC_KEY FUNC_LOC	IFLO-TPLNR	030	Functional location
FLC_MCD FUNC_LOC	IFLO-TPLNR	030	Functional location
FLC_MCD DESCRIPT_T	IFLO-PLTXT	040	FL description
FLRQDAT STRUCT_IND	IFLO-TPLKZ	005	FL structure indicator
FLRQDAT FLC_CATEG	IFLO-FLTYP	001	FL category
FLRQDAT DESCRIPT_T	IFLO-PLTXT	040	FL description
FLRQDAT FLC_REF	IFLO-TRPNR	030	Reference FL
FLRQDAT LOC_PLANT	IFLO-SWERK	004	Maintenance plant
FLRQDAT LOCATION	IFLO-STORT	010	Location
FLRQDAT FLC_ROOMNR	IFLO-MSGRP	008	Room
FLRQDAT COST_CENT	IFLO-KOSTL	010	Cost center
FOLDATA OBJECT_ID	SOFOLENTI1-DOC_ID	046	ID of a folder entry
FOLDATA OBJ_TITLE	SOFOLENTI1-OBJ_DESCR	050	Title of document
FOLDATA SENDER	SOFOLENTI1-SEND_FNAM	035	Full name of sender
FOLDATA IND_EXPRES	SOFOLENTI1-EXPRESS	001	Send express
FOLDATA IND_READ	SOFOLENTI1-READ	001	Object already viewed
FOLDATA IND_REPLY	SOFOLENTI1-TO_ANSWER	001	Object replied to
FOLDATA IND_COPY	SOFOLENTI1-COPY	001	Send as a copy
FOLDATA IND_BLIND	SOFOLENTI1-BLIND_COPY	001	Send as a blind copy
FOLDATA IND_TO_DO	SOFOLENTI1-TO_DO_GRP	001	ToDo group of recipient
ITEMDAT ITEM_NO	RC29P-POSNR	004	BOM item number
ITEMDAT MATERIAL	RC29P-IDNRK	018	BOM component
ITEMDAT DESCRIPT_M	MAKT-MAKTX	040	Material description
ITEMDAT QUANTITY	RC29P-MENGE	018	Component quantity WH_USED
ITEMDAT UNIT	RC29P-MEINS	003	Comp. unit of measure
ITEMDAT ITEM_CATEG	RC29P-POSTP	001	Item category
ITEMDAT SORT_IND	RC29P- SORTF	010	Sort string
ITEMDAT PROD_IND	RC29P-SANFE	001	Item relev. to production
ITEMLIS LEVEL	STPOX-STUFE	002	Level
ITEMLIS PRE_COMP	CSCMAT-MATNR	018	Material number

TCID

ITEMLIS ITEM_NO	STPOX-POSNR	004	BOM item number
ITEMLIS MATERIAL	STPOX-IDNRK	018	BOM component
ITEMLIS DESCRIPT	STPOX-OJTXP	040	Object description (item)
ITEMLIS QUANTITY	STPOX-MNGLG	018	Component quantity WH_USED
ITEMLIS UNIT	STPOX-MEINS	003	Comp. unit of measure
ITEMLIS ITEM_CATEG	STPOX-POSTP	001	Item category
ITEMLIS SORT_IND	STPOX-SORTF	010	Sort string
ITEMLIS PROD_IND	STPOX-SANFE	001	Item relev. to production
MAT_KEY MATERIAL	MARA-MATNR	018	Material number
MTRQDAT MAT_TYPE	MARA-MTART	004	Material type
MTRQDAT INDUSTRY	MARA-MBRSH	001	Industry sector
MTRQDAT DESCRIPT_M	MAKT-MAKTX	040	Material description
MTRQDAT BASE_UNIT	MARA-MEINS	003	Base unit of measure
MTRQDAT BASIC_MAT	MARA-WRKST	014	Basic material
MTRQDAT GROSS_WGHT	MARA-BRGEW	017	Gross weight
MTRQDAT UNIT_WGHT	MARA-GEWEI	003	Unit of weight
MTRQDAT NET_WEIGHT	MARA-NTGEW	017	Net weight
MTRQDAT DIN_FORMAT	MARA-FORMT	004	Page format
MTRQDAT SIZE/DIMEN	MARA-GROES	032	Size/dimension
MTRQDAT MM/PP_STAT	MARC-MMSTA	002	Status from MM/PP view
MTRQDAT PRIZE_AVG	MBEW-VERPR	014	Moving average price
MTRQDAT PRIZE_STD	MBEW-STPRS	014	Standard price
MTRQDAT PLN_DELIV	MARC-PLIFZ	003	Delivery time in days
MTRQDAT AVL_STOCK	MARD-LABST	017	Stock with unrestr. use
MTRQLIS MATERIAL	MARA-MATNR	018	Material number
MTRQLIS MAT_TYPE	MARA-MTART	004	Material type
MTRQLIS INDUSTRY	MARA-MBRSH	001	Industry sector
MTRQLIS DESCRIPT_M	MAKT-MAKTX	040	Material description
MTRQLIS BASE_UNIT	MARA-MEINS	003	Base unit of measure
MTRQLIS MM/PP_STAT	MARC-MMSTA	002	Status from MM/PP view
WH_USED LEVEL	STPOV-LEVEL	002	Level
WH_USED MATERIAL	CSCMAT-MATNR	018	Material number
WH_USED EQUIPMENT	CSCEQUI-EQUNR	018	Equipment number
WH_USED BOM_TYPE	STPOV- BMTYP	001	BOM category
WH_USED DESCRIPT	STPOV-OJTXB	040	Object description (ass.)
WH_USED PLANT	STPOV-WERKS	004	Plant
WH_USED BOM_USAGE	STPOV-STLAN	001	BOM usage
WH_USED ITEM_CATEG	STPOV-POSTP	001	Item category
WH_USED QUANTITY	STPOV-MENGE	018	Component quantity WH_USED
WH_USED UNIT	STPOV-MEINS	003	Comp.unit of measure

TCIU

Set name	Field name	Len	Description
BOCNDAT	MATERIAL	018	Material number
BOCNDAT	PLANT	004	Plant
BOCNDAT	BOM_USAGE	001	BOM usage
BOCNDAT	ALTERNATIV	002	Alternative BOM
BOCNDAT	CHANGE_NO	012	Change number
BOCNNEW	DESCRIPT_B	040	BOM text
BOCNNEW	DESCRIPT_A	040	Alternative BOM text
BOCNNEW	BOM_STATUS	002	BOM status
BOCNNEW	BOM_QUAN	017	Base quantity
BOCRDAT	MATERIAL	018	Material number
BOCRDAT	PLANT	004	Plant
BOCRDAT	BOM_USAGE	001	BOM usage
BOCRDAT	ALTERNATIV	002	Alternative BOM
BOCRDAT	CHANGE_NO	012	Change number
BOCRDAT	DESCRIPT_B	040	BOM text
BOCRDAT	DESCRIPT_A	040	Alternative BOM text
BOCRDAT	BOM_STATUS	002	BOM status
BOCRDAT	BOM_QUAN	017	Base quantity
BORQDAT	MATERIAL	018	Material number
BORQDAT	PLANT	004	Plant
BORQDAT	ALTERNATIV	002	Alternative BOM
BORQDAT	APPLICAT	004	Application
BORQDAT	VALID_FROM	010	Valid-from date
BORQDAT	CHANGE_NO	012	Change number
BORQDAT	REVISION	002	Revision level
BORQDAT	REQ_QUANT	017	Required quantity
BORQDAT	ITEM_CATEG	001	Item category (bill of material)
BORQDAT	ASSEMBLY	018	Assembly
CHGDAT1	DESCRIPT_C	040	Change number description
CHGDAT1	VALID_FROM	010	Valid-from date
CHG_DAT	CHANGE_NO	012	Change number
CHG_DAT	DESCRIPT_C	040	Change number description
CHG_DAT	CHG_STATUS	002	Status of change number
CHG_DAT	VALID_FROM	010	Valid-from date
CHG_DAT	CHG_REASON	040	Reason for change
CHG_DAT	CLASS_TYPE	003	Class type
CHG_DAT	CLASS_NO	018	Class number
CHG_DAT	STD_CLASS	001	Indic: standard class
CHG_DAT	REQUIRED	001	Indic: required characteristic
CHG_DAT	OPTIONAL	001	Indic: display optional char.
CHG_DAT	ASSIGNED	001	Indic: characteristics with values
CHG_DAT	UNASSIGNED	001	Indic: chars without value
CHG_DAT	INHERITED	001	Indic: inherited characteristics
CHG_KEY	CHANGE_NO	012	Change number
DCCRDAT	DOC_NUMBER	025	Document number

TCIU

DCCRDAT	DOC_TYPE	003	Dokument type
DCCRDAT	DOC_PART	003	Document part
DCCRDAT	DOC_VERS	002	Dokument version
DCCRDAT	DESCRIPT_D	040	Description of document
DCCRDAT	DOC_STATUS	002	Status of a document
DCCRDAT	USER	012	Name of user responsible
DCCRDAT	DOC_OFFICE	003	Laboratory/design office
DCCRDAT	CHANGE_NO	012	Change number
DCCRDAT	DISP_APPL	003	Application
DCCRDAT	CARRIER	010	Name of data carrier
DCCRDAT	DOC_PATH	070	Original of document
DCCRDAT	PROTOCOL	020	Document mgmt log field
DCCRDAT	P_DOC_NO	025	Superior document number
DCCRDAT	P_DOC_TYPE	003	Superior document type
DCCRDAT	P_DOC_PART	003	Superior document part
DCCRDAT	P_DOC_VERS	002	Superior document version
DCCRDAT	MATERIAL	018	Material number
DCCRDAT	EQUIPMENT	018	Equipment number
DCCRDAT	FUNC_LOC	030	Functional location
DCCRDAT	CLASS_TYPE	003	Class type
DCCRDAT	CLASS_NO	018	Class number
DCCRDAT	STD_CLASS	001	Indicator: standard class
DCCRDAT	REQUIRED	001	Indic: required chars
DCCRDAT	OPTIONAL	001	Indic: display optional chars
DCCRDAT	ASSIGNED	001	Indic: chars with values
DCCRDAT	UNASSIGNED	001	Indic: chars without values
DCCRDAT	INHERITED	001	Indic: inherited chars
DCRQDAT	DOC_NUMBER	025	Document number
DCRQDAT	DOC_TYPE	003	Dokument type
DCRQDAT	DOC_PART	003	Document part
DCRQDAT	DOC_VERS	002	Dokument version
DCRQDAT	CLASS_TYPE	003	Class type
DCRQDAT	CLASS_NO	018	Class number
DCRQDAT	STD_CLASS	001	Indicator: standard class
DCRQDAT	REQUIRED	001	Indic: required chars
DCRQDAT	OPTIONAL	001	Indic: display optional chars
DCRQDAT	ASSIGNED	001	Indic: chars without values
DCRQDAT	UNASSIGNED	001	Indic: inherited chars
DCRQDAT	INHERITED	001	Indic: inherited chars
DOCHEAD	DOC_TYPE	003	Document type
DOCHEAD	DOC_NAME	012	Document name
DOCHEAD	DOC_TITLE	050	Title of document
DOCHEAD	SORT_FIELD	010	Sort field
DOCHEAD	STANDARD	001	Sensitivity: Standard
DOCHEAD	FUNCTIONAL	001	Sensitivity: Functional
DOCHEAD	CONFIDENT	001	Sensitivity: Confidential
DOCHEAD	NO_CHANGE	001	Document cannot be changed
DOCHEAD	EXP_DATE	010	Expiration date of document
DOCHEAD	PRIORITY	001	SAPoffice: document priority
DOCRECL	RECIPIENT	241	Recipient name

DOCRECL	REC_TYPE	001	Indicator for specifying a recipient
DOCRECL	IND_EXPRES	001	Send: Express
DOCRECL	IND_COPY	001	Send: Copy
DOCRECL	IND_REPLY	001	Send: Reply required
DOCRECL	IND_BLIND	001	Send: Blind copy
DOCRECL	IND_NOPRI	001	Send: No printing
DOCRECL	IND_NO_FWD	001	Send: No forwarding
DOCRECL	IND_OUTBOX	001	Objects are copied to the outbox
DSEARCH	DOC_NUMBER	025	Document number
DSEARCH	DOC_TYPE	003	Document type
DSEARCH	DOC_PART	003	Document part
DSEARCH	DOC_VERS	002	Document version
DSEARCH	USER	012	Name of user responsible
DSEARCH	DEPARTMENT	003	Laboratory/design office
DSEARCH	DOC_STATUS	002	Status of a document
DSEARCH	CHANGE_NO	012	Change number
EQCRDAT	EQUIPMENT	018	Equipment number
EQCRDAT	EQUI_CATEG	001	Equipment category
EQCRDAT	DESCRIPT_E	040	Equipment description
EQCRDAT	ASSET	012	Asset main number
EQCRDAT	COST_CENT	010	Cost center
EQCRDAT	CONST_MON	002	Month of construction
EQCRDAT	CONST_YEAR	004	Year of construction
EQCRDAT	EQUI_DATE	010	Date valid
EQCRDAT	HIGH_EQUI	018	Superior equipment
EQCRDAT	MANUFACTOR	030	Asset manufacturer
EQCRDAT	MANUFACT_C	003	Country of manufacture
EQCRDAT	SERIAL_NUM	030	Manufacturer serial number
EQCRDAT	CLASS_TYPE	003	Class type
EQCRDAT	CLASS_NO	018	Class number
EQCRDAT	STD_CLASS	001	Indicator: standard class
EQCRDAT	REQUIRED	001	Indic: required chars
EQCRDAT	OPTIONAL	001	Indic: display optional chars
EQCRDAT	ASSIGNED	001	Indic: chars with values
EQCRDAT	UNASSIGNED	001	Indic: chars without values
EQCRDAT	INHERITED	001	Indic: inherited chars
EQCRDAT	TECH_IDENT	025	Technical identification number
EQCRDAT	FUNC_LOC	030	Functional location
EQCRDAT	MARK_INST	001	Indicator for installing equipment
EQMDDAT	DESCRIPT_E	040	Equipment description
EQMDDAT	CONST_YEAR	004	Year of construction
EQMDDAT	CONST_MON	002	Month of construction
EQMDDAT	MANUFACTOR	030	Equipment manufacturer
EQMDDAT	MANUFACT_C	003	Country of manufacture
EQMDDAT	ASSET	012	Asset main number
EQMDDAT	SERIAL_NUM	030	Manufacturer serial number
EQMDDAT	COST_CENT	010	Cost center
EQMDDAT	HIGH_EQUI	018	Superior equipment
EQMDDAT	FUNC_LOC	030	Functional location
EQMDDAT	I_D_DATE	010	Install/dismantle date

TCIU

EQMDDAT	MARK_INST	001	Indicator for installing equipment
EQMDDAT	MARK_DISM	001	Indicator for dismantling equi.
EQMDDAT	TECH_IDENT	025	Technical identification number
EQRQDAT	EQUIPMENT	018	Equipment number
EQRQDAT	CLASS_TYPE	003	Class type
EQRQDAT	STD_CLASS	001	Indicator: standard class
EQRQDAT	REQUIRED	001	Indic: required chars
EQRQDAT	OPTIONAL	001	Indic: display optional chars
EQRQDAT	ASSIGNED	001	Indic: chars with values
EQRQDAT	UNASSIGNED	001	Indic: chars without values
EQRQDAT	INHERITED	001	Indic: inherited chars
E_CHG_C	EQUIPMENT	018	Equipment number
E_CHG_C	STD_CLASS	001	Indicator: standard class
E_CHG_C	REQUIRED	001	Indic: required chars
E_CHG_C	OPTIONAL	001	Indic: display optional chars
E_CHG_C	ASSIGNED	001	Indic: chars with values
E_CHG_C	UNASSIGNED	001	Indic: chars without values
E_CHG_C	INHERITED	001	Indic: inherited chars
FLCRDAT	FUNC_LOC	030	Functional location
FLCRDAT	FLC_CATEG	001	Functional location category
FLCRDAT	STRUCT_IND	005	Functional location struct. indic.
FLCRDAT	FLC_COPY	030	Functional location to copy from
FLCRDAT	FLC_REF	030	Reference functional location
FLCRDAT	FLC_SUPER	030	Superior functional location
FLCRDAT	DESCRIPT_T	040	Description of functional location
FLCRDAT	LOC_PLANT	004	Maintenance plant
FLCRDAT	LOCATION	010	Location of the object
FLCRDAT	FLC_ROOMNR	008	Room
FLCRDAT	COST_CENT	010	Cost center
FLCRDAT	CLASS_TYPE	003	Class type
FLCRDAT	CLASS_NO	018	Class number
FLCRDAT	STD_CLASS	001	Indicator: standard class
FLCRDAT	REQUIRED	001	Indic: required chars
FLCRDAT	OPTIONAL	001	Indic: display optional chars
FLCRDAT	ASSIGNED	001	Indic: chars with values
FLCRDAT	UNASSIGNED	001	Indic: chars without values
FLCRDAT	INHERITED	001	Indic: inherited chars
FLCRDAT	EQUI_INST	001	Installation of equipment allowed
FLCRDAT	SINGL_INST	001	Single equipment installation allowed
FLMDDAT	DESCRIPT_T	040	Description of functional location
FLMDDAT	LOCATION	010	Location of the object
FLMDDAT	FLC_ROOMNR	008	Room
FLMDDAT	COST_CENT	010	Cost center
FLMDDAT	EQUI_INST	001	Installation of equipment allowed
FLMDDAT	SINGL_INST	001	Single equipment installation allowed
FLRQDAT	FUNC_LOC	030	Functional location
FLRQDAT	CLASS_TYPE	003	Class type
FLRQDAT	STD_CLASS	001	Indicator: standard class
FLRQDAT	REQUIRED	001	Indic: required chars
FLRQDAT	OPTIONAL	001	Indic: display optional chars

FLRQDAT	ASSIGNED	001	Indic: chars with values
FLRQDAT	UNASSIGNED	001	Indic: chars without values
FLRQDAT	INHERITED	001	Indic: inherited chars
F_CHG_C	FUNC_LOC	030	Functional location
F_CHG_C	STD_CLASS	001	Indicator: standard class
F_CHG_C	REQUIRED	001	Indic: required chars
F_CHG_C	OPTIONAL	001	Indic: display optional char
F_CHG_C	ASSIGNED	001	Indic: chars with values
F_CHG_C	UNASSIGNED	001	Indic: chars without values
F_CHG_C	INHERITED	001	Indic: inherited chars
ITEMDAT	ITEM_NO	004	BOM item number
ITEMDAT	MATERIAL	018	Material number
ITEMDAT	QUANTITY	018	Component quantity WH_USED
ITEMDAT	ITEM_CATEG	001	Item category (bill of material)
ITEMDAT	SORT_IND	010	Sort string
ITEMDAT	SPARE_PART	001	Spare part indicator
MTCRDAT	MATERIAL	018	Material number
MTCRDAT	PLANT	004	Plant
MTCRDAT	STORE_LOC	004	Storage location
MTCRDAT	LVL_ENGIN	001	Indicator: include engineering
MTCRDAT	LVL_CLASS	001	Indicator: include classification
MTCRDAT	LVL_MRP_1	001	Indicator: include MRP 1
MTCRDAT	LVL_MRP_2	001	Indicator: include MRP 2
MTCRDAT	LVL_MRP_3	001	Indicator: include MRP 3
MTCRDAT	LVL_STORE	001	Indicator: include storage
MTCRDAT	LVL_COST	001	Indicator: include costing
MTCRDAT	MAT_TYPE	004	Material type
MTCRDAT	INDUSTRY	001	Industry sector
MTCRDAT	DESCRIPT_M	040	Material description
MTCRDAT	BASE_UNIT	003	Base unit of measure
MTCRDAT	BASIC_MAT	014	Basic material
MTCRDAT	GROSS_WGHT	017	Gross weight
MTCRDAT	UNIT_WGHT	003	Unit of weight
MTCRDAT	NET_WEIGHT	017	Net weight
MTCRDAT	DIN_FORMAT	004	Page format of production memo
MTCRDAT	SIZE/DIMEN	032	Size/dimension
MTCRDAT	MRP_TYPE	002	MRP type
MTCRDAT	MM/PP_STAT	002	Material status from MM/PP view
MTCRDAT	CLASS_TYPE	003	Class type
MTCRDAT	CLASS_NO	018	Class number
MTCRDAT	STD_CLASS	001	Indicator: standard class
MTCRDAT	REQUIRED	001	Indic: Required chars
MTCRDAT	OPTIONAL	001	Indic: display optional chars
MTCRDAT	ASSIGNED	001	Indic: Chars with values
MTCRDAT	UNASSIGNED	001	Indic: chars without values
MTCRDAT	INHERITED	001	Indic: inherited chars
MTRQDAT	MATERIAL	018	Material number
MTRQDAT	PLANT	004	Plant
MTRQDAT	STORE_LOC	004	Storage location
MTRQDAT	LVL_ENGIN	001	Indicator: include engineering

TCIU

MTRQDAT	LVL_MRP_1	001	Indicator: include MRP 1 view
MTRQDAT	LVL_MRP_2	001	Indicator: include MRP 2
MTRQDAT	LVL_MRP_3	001	Indicator: include MRP 3
MTRQDAT	LVL_STORE	001	Indicator: include storage
MTRQDAT	LVL_COST	001	Indicator: include costing
MTRQDAT	LVL_CLASS	001	Indicator: include classification
MTRQDAT	CLASS_TYPE	003	Class type
MTRQDAT	CLASS_NO	018	Class number
MTRQDAT	STD_CLASS	001	Indicator: standard class
MTRQDAT	REQUIRED	001	Indic: Required chars
MTRQDAT	OPTIONAL	001	Indic: display optional chars
MTRQDAT	ASSIGNED	001	Indic: chars with values
MTRQDAT	UNASSIGNED	001	Indic: chars without values
MTRQDAT	INHERITED	001	Indic: inherited chars
M_CHG_C	MATERIAL	018	Material number
M_CHG_C	LVL_ENGIN	001	Indicator: include engineering
M_CHG_C	LVL_CLASS	001	Indicator: include classification
M_CHG_C	LVL_MRP_1	001	Indicator: include MRP 1
M_CHG_C	LVL_MRP_2	001	Indicator: include MRP 2
M_CHG_C	LVL_MRP_3	001	Indicator: include MRP 3
M_CHG_C	LVL_STORE	001	Indicator: include storage
M_CHG_C	LVL_COST	001	Indicator: include costing
M_CHG_C	PLANT	004	Plant
M_CHG_C	STORE_LOC	004	Storage location
M_CHG_C	REQUIRED	001	Indic: Required chars
M_CHG_C	OPTIONAL	001	Indic: display optional chars
M_CHG_C	UNASSIGNED	001	Indic: chars without values
M_CHG_C	ASSIGNED	001	Indic: chars with values
M_CHG_C	INHERITED	001	Indic: inherited chars
PMCRDAT	DESCRIPT_N	040	Short text
PMCRDAT	MAINT_PG	003	Maintenance planning group
PMCRDAT	EQUIPMENT	018	Equipment number
PMCRDAT	FUNC_LOC	030	Functional location
PMCRDAT	CAUSE_CODE	004	Cause code
PMCRDAT	CAUSE_GRP	008	Code group - Causes
PMCRDAT	NOTIF_TYPE	002	Notification type
PMCRDAT	STARTDAT	010	Start of malfunction (date)
PMCRDAT	STARTTIME	008	Start of malfunction (time)
PMCRDAT	MAINTPLANT	004	Planning plant
PMCRDAT	PRIORITY	001	Priority
PMCRDAT	DESCR_POS	040	Notification item short text
PMCRDAT	DESCR_MEAS	040	Task short text
WH_USED	MATERIAL	018	Material number
WH_USED	DIRECT	001	Indicator: direct where-used list
WH_USED	VIA_CLASS	001	Indicator: where-used list via class
WH_USED	MAT_BOM	001	Indicator: material BOM
WH_USED	EQUI_BOM	001	Indicator: equipment BOM
WH_USED	VALID_FROM	010	Valid-from date
WH_USED	VALID_TO	010	Valid-to date
WH_USED	PLANT	004	Plant

WH_USED	ITEM_CATEG	001	Item category (bill of material)
WH_USED	BOM_USAGE	001	BOM usage

Technical Description of the Dialog RFC Interface

This section contains information on how to establish an RFC connection between an SAP System and an external application. This includes information on the settings that you need to make in the customizing functions of your SAP System.

General Information on the Dialog RFC Interface [Seite 455]

How the Dialog RFC Interface Works [Seite 456]

Applications of the Dialog RFC Interface [Seite 457]

Notes on Installation [Seite 458]

Customizing the Dialog RFC Interface [Seite 459]

Configuration File cadrfc.ini [Seite 460]

Configuration File saprfc.ini [Seite 69]

Description of the Available RFC Function Modules [Seite 471]

Description of Dialog RFC Function Library cadrfc00 [Seite 551]

General Information on the Dialog RFC Interface

This section contains information on how the dialog RFC interface works. It also offers an overview of the areas of a company where the dialog RFC interface can be used.

How the Dialog RFC Interface Works [Seite 456]

Applications of the Dialog RFC Interface [Seite 457]

How the Dialog RFC Interface Works

With the dialog RFC interface, you can call SAP dialog transactions directly from a technical application.

- Once the connection is established, RFC function calls can be made from the external system. Since the connection is established via an active GUI session, these function calls can also contain SAP dialog screens.

  A whole series of function modules that can be called by *Remote Function Call* has been developed especially for the CAD interface. Most of these modules allow you to access SAP dialog transactions easily, and to transfer data.

  If required, you can also use all the function calls and API function calls of the CAD interface directly.

  You can establish the connection to the SAP R/3 System directly from the external application with the function *CadRfcOpen*. A SAPGUI will not be started with MS Windows 3.1. You can only call functions that do not process SAP dialog screens. You can select the R/3 System you require either via configuration file *cadrfc.ini* or by entering the application server and the R/3 System ID directly (see also configuration file *cadrfc.ini* and the documentation for API *CadRfcOpen*).

Applications of the Dialog RFC Interface

RFC function modules are supported for the following application areas:

- Document management
- Material master records
- Bills of material
- Change master record
- SAP mail and workflow
- Plant maintenance

  (equipment master records, functional locations, maintenance notifications, and maintenance orders)
- General Functions

  (find objects via matchcode, find objects via classes, SAP system parameters, product structure, where-used list)

See also:

Description of the Available RFC Function Modules [Seite 471]

Notes on Installation

The workstation-specific part of the software is available as standard for the following operating systems:

- DEC OSF/1
- HP UX
- IBM AIX
- SNI SINIX
- Sun Solaris
- MS Windows 3.1
- MS Windows NT
- MS Windows95 (software identical to MS Windows NT)

You will find the software on the SAP CD *Presentation with SAP-GUI*.

For Unix platforms, the software is stored as a compressed file called *cad.car* in the relevant platform sub-directories. You can decompress the file using the SAP utility program *car*.

For MS Windows 3.1, MS Windows NT and MS Windows95, you can install the software using an additional option of the SAP GUI installation program.

After installation, see the file *cadread.me* for information on the files supplied, as well as the latest information on the current version of the interface.

To install the software in your own applications, you also require the CPIC or RFC library to communicate with R/3. These are a component part of the RFC SDK, and can also be found on the SAP CD *Presentation with SAP-GUI*.

Customizing the Dialog RFC Interface

You can make the following company-specific settings in the R/3 customizing functions for the CAD interface:

- Configuration File saprfc.ini [Seite 69]
- Configuration File cadrfc.ini [Seite 460]

Configuration File cadrfc.ini

The following types of information are stored in this file:

- Parameters for logging on to the SAP R/3 System

  CadRfcUser

  SAP user name that you use when you log on to the SAP R/3 System via the interface. The SAP user must exist in the SAP R/3 System and have the required authorizations. If no parameter has been set, you must transfer the parameter when you call function *SapConnc* or *CadRfcOpen.*

  CadRfcPassword

  User password in the SAP R/3 System. If no parameter has been set, you must transfer the parameter when you call function *SapConnc* or *CadRfcOpen.*

  CadRfcClient

  SAP client that the user wishes to work in. If no parameter has been set, you must transfer the parameter when you call function *SapConnc* or *CadRfcOpen.*

  CadRfcLanguage

  Logon language for the SAP R/3 System. If no parameter has been set, you must transfer the parameter when you call function *SapConnc* or *CadRfcOpen.*

- **Default values for establishing a connection**

  The following parameters are only relevant if you establish a connection to the R/3 System from your external application.

  CadRfcDestination

  If you want the target system to be determined from configuration file *saprfc.ini,* you can enter the logical name of the R/3 partner system here. A matching entry must exist in file *saprfc.ini.*

  CadRfcHostname and CadRfcSysnr

  Here, you can define default values for the application server and system ID of your chosen R/3 System. In this case, configuration file *saprfc.ini* is no longer required.

  CadRfcUser, CadRfcPassword, CadRfcClient, and CadRfcLanguage

  Here, you can define default values for logging on to the R/3 System (refer to the documentation on API *CadRfcOpen).*

  CadRfcMessageServer

  If you enter a message server, the connection to the SAP System can be established using an application server group. You must also enter the logical SAP System name as parameter *CadRfcDestination.* In this case, file *saprfc.ini* is not required. Parameter *CadRfcGroup* determines which application server group to use.

  CadRfcUseSapMsgFile

  If you enter the letter “Y”, the message server is automatically determined from file *sapmsg.ini.* When you install the SAPGUI, this file is copied to the Windows directory (Windows 3.11, NT, and 95). If you are using UNIX, you can copy this file to your working

directory, or you must set environment variable RFC_LOGON_INI_PATH. This function is switched off if you enter the letter “N”.

CadRfcGroup

Here you can enter the application server group that is to be used when you establish a connection using a message server. If you do not set this parameter, group PUBLIC is used.

- **Dialog interface parameters**

  CadDialogSapVersion

  If you are using the function SapConnc, the CAD interface enables you to establish a connection to an R/2 or an R/3 System. You can use this parameter to choose the SAP version. You can choose between: R/2 and R/3.

  CadDialogCadSystem

  The SAP-CAD interface can work in parallel with other external systems. If necessary, you can specify for each system which data is transferred with each function. (For further information on this, refer to the online documentation on allocating field sets in Customizing for the CAD interface.) This parameter specifies which configuration is used in the SAP System. If no parameter is set here, it must be specified when you call the function CONNC.

  CadDialogNetAddress

  If you want to use document management functions to access the storage locations of documents, you must identify the frontend computer. In the standard system, this is done by using environment variable HOSTNAME with a PC and environment variable DISPLAY with Unix (for more information, refer to the documentation on the document management system). If you do not use these environment variables, you can use parameter *CadDialogNetAddress* to define a frontend computer.

  CadDialogEmulationMode

  If you want to work with CAD dialog interface emulation, you must set this parameter to 1.
  You can make the following entries:

  - Working with emulation
  - Working without emulation

  Additionally, you must either copy cadcpemu.dll to caddialg.dll, or you must link cadcpemu.dll to the emulation library.

- **Trace parameters**

  CadRfcTrace

  This parameter switches the extended RFC trace on and off. The RFC library in the SAP function library has its own trace routines. Detailed trace information is stored in the trace. This trace is of most use when there are problems with establishing a connection, or if the connection terminates unexpectedly. The trace file is usually created in the program start directory. You can use parameter *CadTraceDir* to write all trace files to another directory.

  The following trace files are created:

- CPICXXXXX.TRC (XXXXX for an allocated process ID)
- RFC_XXXXX.TRC (RFC trace with Windows)
- CAD_RFC.LOG (Dialog RFC interface trace with Windows)
- DEV_RFC (RFC trace and dialog RFC trace with UNIX)

You can make the following entries:

1	Trace is switched on.
0	Trace is switched off.

CadDialogStandardTrace

This parameter switches the standard trace on and off. In the standard trace, Hex and ASCII format are used to log which data has been exchanged between the workstation and the SAP System. Unless otherwise defined in the *CadTraceDir* parameter, the trace file is created in the program start directory. The trace file name is: log_xxxx.log, where xxxx stands for the allocated process ID.

You can make the following entries:

1	Trace is switched on.
0	Trace is switched off.

CadDialogSapCadTrace

This parameter switches a special trace on and off. This special trace can be used as input for interface emulation if no R/3 System is available. In this case, the contents of the log file must be copied to a file with the name *cademu.r3*. This file then serves as input for the emulation.

Unless otherwise defined in the *CadTraceDir* parameter, the trace file is created in the program start directory. The trace file name is: sap_cad.log.

You can make the following entries:

1	Trace is switched on.
0	Trace is switched off.

CadTraceDir

All trace files are written to the directory entered in this parameter. The environment variables CPIC_TRACE_DIR and RFC_TRACE_DIR are set internally for this purpose

- **ABAP debug parameter**

  CadRfcAbapDebug

  If you set this parameter to 1, the ABAP debugger is called when you call an SAP function for the first time (only with an R/3 System).

  You can make the following entries:

1	ABAP debugger is switched on.
0	ABAP debugger is switched off.

- **Parameters for establishing a connection from the SAP System**

  CadRfcTempDir

  In parameter *CadRfcTempDir,* the path that is defined in Customizing step *Define data carrier types* for document management must be entered in field *Path*. Temporary data is stored under this path.

  You only need to enter this path if you establish the connection from your SAP System.

  CadRfcRouter

  You only need to enter this parameter if you establish a connection to the external application from your SAP System.

  If you access the R/3 System via a router, you must enter the router name here.

- **Default values to be transferred to the SAP System in RFC function calls**

  Here, you can define default values for those parameters in RFC modules in the interface that are seldom or never changed.

  These parameters are only used if the CAD function library for the dialog RFC interface is in use.

You can define default values for the following applications:

Default Values for BOMs [Seite 464]

Default Values for Document Management [Seite 467]

Default Values for Material Master Records [Seite 468]

Default Values for Plant Maintenance [Seite 469]

General Default Values [Seite 470]

Default Values for BOMs

BOM item (create and change)

Default values	Field description
DefaultCsrfcitemAusch	Percentage component scrap
DefaultCsrfcitemBeikz	Material provision indicator
DefaultCsrfcitemErskz	Spare part indicator
DefaultCsrfcitemFmeng	Fixed quantity
DefaultCsrfcitemMeins	Component unit of measure
DefaultCsrfcitemPostp	Item category
DefaultCsrfcitemPotx1	BOM item text (line 1)
DefaultCsrfcitemPotx2	BOM item text (line 2)
DefaultCsrfcitemRvrel	Relevance to sales indicator
DefaultCsrfcitemSanfe	Indicator: item relevant to production
DefaultCsrfcitemSanin	Indicator: item relevant to plant maintenance
DefaultCsrfcitemSanka	Indicator: item relevant to costing
DefaultCsrfcitemSanko	Indicator: item relevant to engineering/design
DefaultCsrfcitemSanvs	Indicator: item relevant to shipping
DefaultCsrfcitemSchgt	Indicator: bulk material
DefaultCsrfcitemSortf	Sort string
DefaultCsrfcitemStkkz	Indicator: PM assembly
DefaultCsrfcitemStlal	Alternative BOM
DefaultCsrfcitemEwahr	Percentage usage probability
DefaultCsrfcitemAlpg	Alternative item group
DefaultCsrfcitemDspt	Explosion type
DefaultCsrfcitemPrvbe	Supply area
DefaultCsrfcitemNfeag	Discontinuation group
DefaultCsrfcitemNfgrp	Follow-up group
DefaultCsrfcitemKzkup	Indicator: co-product
DefaultCsrfcitemIntrm	Intra material
DefaultCsrfcitemKzclb	Indicator: Classification as selection condition

BOM sub-item (create and change)

Default values	Field description

DefaultCssubitemUpmng	Sub-item quantity
DefaultCssubitemUptxt	Sub-item text

BOM header (create and change)

Default values	Field description
DefaultBicskAennrCreate	Change number for creating BOM
DefaultBicskAennrChange	Change number for changing BOM
DefaultBicskBmein	Base unit of measure
DefaultBicskBmeng	Base quantity
DefaultBicskDatub	Valid-to date
DefaultBicskDatuv	Valid-from date
DefaultBicskEmeng	Required quantity
DefaultBicskExstl	BOM group
DefaultBicskLabor	Department
DefaultBicskLosbs	From lot size
DefaultBicskLosvn	To lot size
DefaultBicskStktx	Alternative BOM text
DefaultBicskStlal	Alternative BOM
DefaultBicskStlan	BOM application
DefaultBicskStlbe	Authorization group
DefaultBicskStlst	BOM status
DefaultBicskWerks	Plant
DefaultBicskZtext	BOM header text
DefaultBicskRevlv	Revision level

BOM (display)

Default values	Field description

Default Values for BOMs

DefaultBomSkipFirstScreen	Indicator: skip initial screen
DefaultBomAction	Display function
DefaultBomAlternative	Alternative BOM
DefaultBomApplication	BOM application
DefaultBomChangeNumber	Change number
DefaultBomRevisionLevel	Revision level
DefaultBomType	BOM usage
DefaultBicskLabor	Laboratory/design office

Default Values for Document Management

Default values	Field description
DefaultDocDokar	Document type
DefaultDocDoktl	Document part
DefaultDocDokvr	Document version
DefaultDocDoknrVl	Copy from document number
DefaultDocDoktlVl	Copy from document part
DefaultDocDokvrVl	Copy from document version
DefaultDocDokst	Status
DefaultDocProtf	Log field for status
DefaultDocLabor	Department
DefaultDocAennr	Change number
DefaultDocPrenr	Document (superior)
DefaultDocPretl	Document part (superior)
DefaultDocPrear	Document type (superior)
DefaultDocPrevr	Document version(superior)
DefaultDocBegru	Authorization group
DefaultDocDttrg	Data carrier 1
DefaultDocDappl	Application 1
DefaultDocDttrg1	Data carrier 2
DefaultDocDappl1	Application 2
DefaultDocClassType	Class type
DefaultDocClassNo	Class

Default Values for Material Master Records

Default values	Field description
DefaultMatChangeNumber	Change number
DefaultMatApplication	Material category
DefaultMatType	Material type
DefaultMatRefMaterial	Copy from material number
DefaultMatClassType	Class type
DefaultMatClassNo	Class

Default Values for Plant Maintenance

Equipment master record

Default values	Field description
DefaultEquClassType	Class type
DefaultEquType	Equipment type

Functional location

Default values	Field description
DefaultFloClassType	Class type
DefaultFloStructIndicator	Structure indicator
DefaultFloType	Functional location category

Maintenance processing

Default values	Field description
DefaultNotType	Notification type

Maintenance order

Default values	Field description
DefaultPmOrderType	Order type

General Default Values

Default values	Field description
DefaultValidFrom	Valid-from date
DefaultValidTo	Valid-to date
DefaultPlant	Plant
DefaulstBusinessArea	Business area

Description of the Available RFC Function Modules

This section describes the function modules that are made available by the CAD dialog RFC interface functionality.

You can use example program *cadrftst*, also supplied as a C source file with the workstation software, to test the interface. You can also use this program as a basis for your own developments. You can establish a connection to the external system from your R/3 System. As of release 3.0C, you can also establish a connection directly from the example program.

Transfer Tables SAP_FIELD_DATA and DMS_CLASS_DATA [Seite 472]

Inserting and Checking Out Documents [Seite 486]

Functions for Document Management [Seite 473]

Functions for Material Master Records [Seite 489]

Functions for Engineering Change Management [Seite 498]

Mail and Workflow Functions [Seite 536]

BOM Functions [Seite 504]

General Functions in the SAP Environment [Seite 540]

Displaying a Product Structure [Seite 541]

Displaying a Where-Used List for a Material or Document [Seite 543]

Finding Objects via Matchcode [Seite 545]

Finding Objects in Classes [Seite 547]

Requesting SAP System Parameters [Seite 550]

Functions for Plant Maintenance [Seite 512]

Transfer Tables SAP_FIELD_DATA and DMS_CLASS_DATA

In a number of function calls, tables *SAP_FIELD_DATA* and *DMS_CLASS_DATA* are also transferred. These two tables are not currently used, and are reserved for future enhancements to the interface.

Functions for Document Management

The following document management records are supported:

Creating a Document Info Record [Seite 474]
Changing a Document Info Record [Seite 476]
Displaying a Document Info Record [Seite 478]
Finding a Document Info Record [Seite 480]
Inserting a Document Info Record [Seite 482]
Reading a Document Info Record [Seite 484]
Inserting and Checking Out Documents [Seite 486]
Reading Data for the Drawing Header [Seite 487]

Creating a Document Info Record

You can use this function module to create a document info record in the SAP System.

Module name:	RFC_CREATE_DOCUMENT_MASTER
C prototype:	*int CadRfcDocCreate(DRAW_RFC *, CadRfcImportField *, ITAB_H**, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: DOCUMENT_DATA	 Structure	 DRAW_RFC	 Structure with document data
Export parameters: FUNCTION_CODE EXP_DOCUMENT_DATA	 CHAR Structure	 14 DRAW_RFC	 Function code Structure with document data
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Object link Not currently in use

You can transfer object links in the table SAP_FIELD_DATA by using the following data:

FIELDNAME	Name of the object type (data base table, such as MARA)
OBJECTKEY	Identified object data (such as material number)

Import parameters

Structure DOCUMENT_DATA

Structure *DOCUMENT_DATA* can be used to transfer all the essential fields of the document info record.

The following fields are not transferred:

- Data on object links
- Classification data
- Descriptions in languages other than the logon language

You can call the precise structure of base structure DRAW_RFC via the SAP Dictionary (R/3 Repository).

If all four key fields of the document are filled and the document does not yet exist, the initial screen of the transaction for creating a document is processed in the background and you see the main document screen directly.

All the usual fields are transferred. The interface ensures that all the necessary checks are made.

Special feature of field *DOCUMENT_DATA-PATHNAME*

Field *DOCUMENT_DATA-PATHNAME* is a special feature. This field can contain a complete file path with file name (for example, D:\DATA\CAD\SCREW1.DWG). If you do this, the system tries to determine a valid data carrier and original application file name, using the Customizing settings for the Document Management System.

For example, if data carrier *ServerCad* is defined, with the prefix *D:\DATA\CAD*, the value *ServerCad* is transferred to DMS field *Data carrier* and the value *SCREW1.DWG* is transferred to field *Original name*.

The following rules apply:

Data carrier type 1	Data carrier type 2	Storage data
Set	Not set	Store in original 1
Set	Set	Store in original 1
Not set	Set	Store in original 2
Not set	Not set	Store in original 1

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_CO	Document created and checked out
DOC_UPDATE	Document created
Space	No changes to data

Before you can process original application data with function code *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING*.

EXP_DOCUMENT_DATA

If function codes *DOC_CO* and *DOC_UPDATE* are returned, the document data has been created. The created document data is made available in structure *EXP_DOCUMENT_DATA*. If no document data has been saved, the structure contains initial values.

A document info record is only created if function code *DOC_UPDATE* is set and a valid document key that has not been used for an existing document is entered.

Otherwise, the document key must be checked via the relevant fields of structure *EXP_DOCUMENT_DATA*.

Changing a Document Info Record

You can use this function module to change an existing document info record in the SAP System.

Module name:	RFC_CHANGE_DOCUMENT_MASTER
C prototype:	*int CadRfcDocChange(DRAW_RFC *, CadRfcImportField *, ITAB_H**, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
DOCUMENT_DATA	Structure	DRAW_RFC	Structure with document data
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_DOCUMENT_DATA	Structure	DRAW_RFC	Structure with document data
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

Structure DOCUMENT_DATA

Structure *DOCUMENT_DATA* can be used to transfer all the essential fields of the document info record.

The following fields are not transferred:

- Data on object links
- Classification data
- Descriptions in languages other than the logon language

You can call the precise structure of base structure DRAW_RFC via the SAP Dictionary (R/3 Repository).

If all four key fields of the document are filled, the initial screen of the transaction for displaying a document is processed in the background and you see the main document screen directly (provided that no error occurs).

All the usual fields are transferred. Existing data may be overwritten. The interface ensures that all the necessary checks are made.

Special feature of field *DOCUMENT_DATA-PATHNAME*

Field *DOCUMENT_DATA-PATHNAME* is a special feature. You can enter a complete file path with file name in this field (for example D:\DATA\CAD\SCREW1.DWG). If you do this, the system tries to determine a valid data carrier and original application file name, using the customizing settings for the document management system. This data is then transferred to the relevant document management fields.

The following rule applies:

Data carrier type 1	Data carrier type 2	Storage location
Set	Not set	Store in original 1
Set	Set	Store in original 1
Not set	Set	Store in original 2
Not set	Not set	Store in original 1

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	No change to data, but request to insert a document
DOC_CO	Document changed and checked out
DOC_UPDATE	Document changed
Space	no changes to data

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING*.

Struktur EXP_DOCUMENT_DATA

If function codes *DOC_CO* and *DOC_UPDATE* are returned, the document data has been changed. The changed document data is made available in structure *EXP_DOCUMENT_DATA*. If no changes to document data have been saved, the structure contains initial values.

A document info record is only changed if function code *DOC_UPDATE* is set and a valid document key is entered. Otherwise, the document key must be checked via the relevant fields of structure *EXP_DOCUMENT_DATA*. Function *DOC_CO* is especially designed such that a new version can be created from an existing released version, and then checked out immediately.

Displaying a Document Info Record

You can use this function module to display an existing document info record in the SAP System.

Module name:	RFC_DISPLAY_DOCUMENT_MASTER
C prototype:	*int CadRfcDocDisplay(CadRfcExportField *, CadRfcImportField *,ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
DOC_NUMBER	CHAR	25	Document number
DOC_TYPE	CHAR	3	Document type
DOC_PART	CHAR	2	Document part
DOC_VERS	CHAR	2	Document version
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

DOC_NUMBER, DOC_TYPE, DOC_PART, DOC_VERS

If all four key fields of the document are filled, the initial screen of the transaction for displaying a document is processed in the background and you see the main document screen directly (provided that no error occurs).

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
DOC_UPDATE	Document changed
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

In the document management system, you can switch from display mode to change mode at any time. This means that you can change the document info record under certain circumstances. The changed document data can be requested with function call *RFC_INSERT_DRAWING* if *DOC_UPDATE* is used.

The document you enter here need not be the same as the document you entered initially. This is because, when you display a document, you always have the option of creating a new version directly or going to other documents via the object links.

Finding a Document Info Record

You can use this function module to find document info records in the SAP System.

Module name:	RFC_SELECT_DOCUMENT_MASTER
C prototype:	*int CadRfcDocSelect(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

You can branch from the list of documents to other functions as required. You can select one of the found documents at any time in order to do an insert or a check out.

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
DOC_NUMBER	CHAR	25	Document number
DOC_TYPE	CHAR	3	Document type
DOC_PART	CHAR	2	Document part
DOC_VERS	CHAR	2	Document version
USER_NAME	CHAR	12	User name
CHANGE_NUMBER	CHAR	12	Change number
OFFICE	CHAR	3	Laboratory/ design office
DOCUMENT_STATUS	CHAR	2	Document status
SKIP_FIRST_SCREEN	CHAR	1	Flag: Skip initial screen
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

DOC_NUMBER, DOC_TYPE, DOC_PART, DOC_VERS

You can predefine the document key fields, either exactly or generically.

USER_NAME, CHANGE_NUMBER, OFFICE, DOCUMENT_STATUS

You can restrict the result of your search by making entries in these fields. For example, you can send a query such as: show me all the documents that I have processed. In the USER_NAME field, you can make a generic entry.

SKIP_FIRST_SCREEN

If this parameter is “X”, you skip the initial screen of the find document transaction and go directly to the resulting list.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

Inserting a Document Info Record

This function module makes the document key and the access path to the original application file available for a document to be inserted or for a checked out document.

Module name:	RFC_INSERT_DRAWING
C prototype:	*int CadRfcDocInsert(DRAW_RFC *, CadRfcImportField *, ITAB_H**,ITAB_H **)*

You can select or check out a document to be inserted in another application from many of the available functions. If you do this, either the value *DOC_INSERT* (request to insert a document) or *DOC-CO* (document changed and checked out) is set in the return parameter FUNCTION_CODE. To load the key fields of the selected document and the access path to the original application file, you then call function module *RFC_INSERT_DRAWING*.

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: None			
Export parameters: FUNCTION_CODE EXP_DOCUMENT_DATA	 CHAR Structure	 14 DRAW_RFC	 Function code Structure with document data
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Not currently in use Not currently in use

Export parameters

Struktur EXP_DOCUMENT_DATA

The document key of the selected document comprises the following four fields:

EXP_DOCUMENT_DATA-DOKNR (Document number)

EXP_DOCUMENT_DATA-DOKAR (Document type)

EXP_DOCUMENT_DATA-DOKTL (Document part)

EXP_DOCUMENT_DATA-DOKVR (Document version)

The file access path is made availabe in the field EXP_DOCUMENT_DATA-PATHNAME.

The other fields are only filled out if the document has been checked out. In this case, all the essential fields of the document info record are shown.

The following fields are not transferred:

- Data on object links
- Classification data
- Descriptions in languages other than the logon language

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

NOT_FOUND	No data found to insert
Space	Other

Reading a Document Info Record

You can use this function to request the master data for the document you enter.

Module name:	RFC_GET_DMS_DATA
C prototype:	*int CadRfcGetDmsData(CadRfcExportField *, DRAW_RFC *,CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
DOC_NUMBER	CHAR	25	Document number
DOC_TYPE	CHAR	3	Document type
DOC_PART	CHAR	2	Document part
DOC_VERS	CHAR	2	Document version
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_DOCUMENT_DATA	Structure	DRAW_RFC	Structure with document data
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

DOC_NUMBER, DOC_TYPE, DOC_PART, DOC_VERS

Document key

Export parameters

Struktur EXP_DOCUMENT_DATA

Structure *DOCUMENT_DATA* contains all the essential fields of the document info record.

The following fields are not transferred:

- Data on object links
- Classification data
- Descriptions in languages other than the logon language

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

NOT_FOUND	No data found to insert
Space	Other

Inserting and Checking Out Documents

If you want to use the SAP document management functionality to manage documents created with external applications, the two applications should be closely integrated. One of the strengths of the SAP document management system is that it allows you to create links to a wide range of SAP objects, such as material master records, change master records, equipment master records, and functional locations.

When you want to find a document, you can often take advantage of the option of using an SAP object to find the documents allocated to this object. The functionality for finding objects via a matchcode or via classes are useful tools for this purpose. For this reason, many of the function modules described allow you to either select a document to insert or request a document to change (check out) directly from the object. The leading application receives the data in the relevant function code and can then call the document key and access path from the SAP System. Files that are stored in an archive or vault subject and managed by the SAP System are first stored in a temporary file on the frontend computer.

Reading Data for the Drawing Header

You can use this function to read data for the drawing header, such as document data or data for the change number.

Module name:	CAD_GET_TITLE_BLOCK_DATA

This function module can be used with any Release from 3.1H to 4.6A by installing a service package (until Release 4.5B so-called "hot packages") in the customer system.
The required information can be found in note 0160527 *CAD Read drawing header data* (component CA-CAD *CAD Integration*).

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Export parameters:			
DOCUMENTDATA	Structure	BAPI_DOC_DRAW	Document data
ECMDATA	Structure	AENR_API02	Change master record
E_RETURN	Structure	CAD_RETURN-VALUE	Return value (0 = ok; 4 = error)
E_MESSAGE	Structure	MESSAGE-MSGTX	Error message
Tables:			
IMPORT_DATA	Structure	RFCDMSDATA	Import data (document key / material key)
OBJECTLINKS	Structure	BAPI_DOC_DRAD	Object links
DOCUMENTDESCRIPTIONS	Structure	BAPI_DOC_DRAT	Language-dependent Document description
LONGTEXTS	Structure	BAPI_DOC_TEXT	Language-dependent Long texts
STATUSLOG	Structure	CAD_DOC_DRAP	Status log
CLASS_DATA	Structure	CLS_CHARAC	Classification data
EXPORT_DATA	Structure	RFCDMSDATA	Material, equipment and functional location data
USER_DATA	Structure	RFCDMSDATA	Customer-specific data

The function module returns the following data:

Object	Data
Document data	Object links of a document
	Language-dependent short texts of a document
	Language-dependent long texts of a document
	Status log entries
	Characteristic value assignment of a document (classification)
	Data for change numbers
	Revision level

Material data	
Equipment data	
Data from functional locations	

You can use a Customer Exit to set fields for a customer.

Functions for Material Master Records

The following functions are supported for material master records:

Creating a Material Master Record [Seite 490]

Changing a Material Master Record [Seite 492]

Displaying a Material Master Record [Seite 494]

Creating a Material Master Record

You can use this function to create a material master record.

Module name:	RFC_CREATE_MATERIAL_MASTER
C prototype:	*int CadRfcMatCreate(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
IMP_MATERIAL	CHAR	18	Material number
IMP_APPLICATION	CHAR	1	Material type
IMP_TYPE	CHAR	4	Material type reference
IMP_REF_MATERIAL	CHAR	18	Material number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_MATERIAL	CHAR	18	Material number
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

IMP_MATERIAL, IMP_APPLICATION, IMP_TYPE

If you enter a material number, the initial screen is processed in the background (provided that no error occurs).

This function is not yet supported in release 3.0A (transfer of material number is not possible).

IMP_REF_MATERIAL

You also have the option of entering the material number of a material you want to copy from.

Export parameters

EXP_MATERIAL

This field contains the material number assigned.

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
UPDATE	Material master record created
Space	Other

Before you can process original application data with function code *DOC_INSERT,* you need to call function module *RFC_INSERT_DRAWING*.

If you link a material to a document as soon as you change the material, you can branch from the material master maintenance function to the document management system. If you request a document to insert, the material master maintenance function you started is terminated, and any data you entered will be lost.

Changing a Material Master Record

You can use this function to change an existing material master record.

Module name:	RFC_CHANGE_MATERIAL_MASTER
C prototype:	*int CadRfcMatChange(CadRfcExportField *, CadRfcImportField *,ITAB_H **, TAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
MATERIAL CHANGE_NUMBER	CHAR CHAR	18 12	Material number Change number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

MATERIAL

If you enter a material number, the initial screen is processed in the background (provided that no error occurs). This happens regardless of whether you enter a change number or not.

CHANGE_NUMBER

You also have the option of entering a change number.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
UPDATE	Material master record changed
Space	Other

Before you can process original application data with function code *DOC_INSERT,* you need to call function module *RFC_INSERT_DRAWING*.

If you link a material to a document as soon as you change the material, you can branch from the material master maintenance function to the document management system. If you request a document to insert, the material master maintenance function you started is terminated, and any data you entered will be lost.

Displaying a Material Master Record

You can use this function to display an existing material master record.

Module name:	RFC_DISPLAY_MATERIAL_MASTER
C prototype:	*int CadRfcMatDisplay(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
MATERIAL	CHAR	18	Material number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Table with material master data
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

MATERIAL

If you enter a material number, the initial screen is processed in the background (provided that no error occurs).

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

Tables

SAP_FIELD_DATA

All fields for the material master structure MARA and the description (MAKT – MAKTX) are returned in this table.

The fields in table SAP_FIELD_DATA are filled as follows:

FIELDMULTI	Language-dependent description of the database field
FIELDNAME	Field name (for example, MARA – MATNR)
FIELDVALUE	Value

You can read an entry in the table SAP_FIELD_DATA with the function CadRfcGetFieldTabEntry.

Reading a Material Master Record

You can use this function to read the master data for a material:

Module name:	RFC_GET_MATERIAL_DATA
C prototype:	*int CadRfcGetMatData (CadRfcExportField *, CadRfcImportField *,ITAB_H **FieldData, ITAB_H **DmsClassData);*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
I_MATERIAL	CHAR	18	Material number
Export parameters:			
E_RETURN	CHAR	10	Return code (0 or 99)
E_MESSAGE	CHAR	200	Error message
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Table with material master data
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

I_MATERIAL Material number

Export parameters

E_RETURN	Return code 0 OK 99 Error
E_MESSAGE	If there are errors, an error message will be returned in this parameter. In this case, the return code (E_RETURN) is 99.

Tables

SAP_FIELD_DATA

All fields for the material master structure MARA and the description (MAKT – MAKTX) are returned in this table.

The fields in table SAP_FIELD_DATA are filled as follows:

FIELDMULTI Language-dependent description of the database field

FIELDNAME Field name (z.B. MARA MATNR)

FIELDVALUE Value

You can read an entry in the table SAP_FIELD_DATA with the function CadRfcGetFieldTabEntry.

Functions for Engineering Change Management

The following functions for change master records are supported:

Creating a Change Master Record [Seite 499]

Changing a Change Master Record [Seite 501]

Displaying a Change Master Record [Seite 503]

Creating a Change Master Record

You can use this function to create a change master record.

Module name:	RFC_CREATE_CHANGE_MASTER
C prototype:	*int CadRfcChgCreate(CadRfcExportField *, CadRfcImportField *,ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
IMP_CHANGE_NO	CHAR	12	Change number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_CHANGE_NO	CHAR	12	Change number
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

IMP_CHANGE_NO

If you enter a change number, the initial screen is processed in the background (provided that no error has occurred).

Export parameters

EXP_CHANGE_NO

This field contains the assigned change number.

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
UPDATE	Material master record changed
Space	Other

Before you can process original application data with function code *DOC_INSERT,* you need to call function module *RFC_INSERT_DRAWING*.

Creating a Change Master Record

If you link a change master record to a document as soon as you create the change master record, you can branch from the change master maintenance function to the document management system. If you request a document to insert, the change master maintenance function you started is terminated, and any data you entered will be lost.

Changing a Change Master Record

You can use this function to change a change master record.

Module name:	RFC_CHANGE_CHANGE_MASTER
C prototype:	*int CadRfcFloChange(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
CHANGE_NO	CHAR	12	Change number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

CHANGE_NO

If you enter a change number, the initial screen is processed in the background (provided that no error has occurred).

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
Space	Other

Before you can process original application data with function code *DOC_INSERT,* you need to call function module *RFC_INSERT_DRAWING*.

If you link a change master record to a document as soon as you create the change master record, you can branch from the change master maintenance function to the document management system. If you request a document to insert, the change master maintenance function you started is terminated, and any data you entered will be lost.

Displaying a Change Master Record

You can use this function to display a change master record.

Module name:	RFC_DISPLAY_CHANGE_MASTER
C prototype:	*int CadRfcChgDisplay(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
CHANGE_NO	CHAR	12	Change number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

CHANGE_NO

If you enter a change number, the initial screen is processed in the background (provided that no error has occurred).

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

Functions for BOMs

The following functions are supported for BOMs:

Creating a Material BOM [Seite 505]

Changing a Material BOM [Seite 507]

Displaying a Material BOM [Seite 509]

Creating a Material BOM

You can use this function to create a material BOM in the SAP System.

Module name:	RFC_CREATE_BILL_OF_MATERIAL
C prototype:	*int CadRfcBomCreate(BICSK *, CadRfcImportField *, ITAB_H **, ITAB_H **)* *int CadRfcWriteBomTabEntry(char*, char*, int)*

First, this function module goes to the initial BOM maintenance screen. All parameters of the initial screen that were transferred with the function call are already filled out. As a security mechanism, it is no longer possible to change the values in the fields *Material number*, *Plant*, *Usage,* and *Alternative* that were transferred via the interface.

All transferred item data is transferred to the BOM. You see the item overview screen of the BOM. You can change the BOM before you save.

A CAD indicator is set in the BOM header and in all items transferred from the external system. This does not apply to new items that were entered in the transaction before the BOM was saved.

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
MATERIAL_HEADER_DATA	Structure	BICSK	BOM header data
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_MAT_HEADER_DATA	Structure	BICSK	Changed BOM header data
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use
DMS_CLASS_DATA	Structure	CSRFCITEM	BOM item data

Import parameters

MATERIAL_HEADER_DATA

In this structure, you can transfer all BOM header data that can be maintained online. You can call the precise structure of base structure *BICSK* via the SAP Dictionary (R/3 Repository).

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

UPDATE	BOM created
Space	Other

EXP_MAT_HEADER_DATA

If a BOM has been changed, the current BOM header data is made available in structure *EXP_MAT_HEADER_DATA*.

Tables

DMS_CLASS_DATA

Table structure *ITEM_DATA* can be used to transfer BOM item data. You can call the precise structure of base structure *CSRFCITEM* via the SAP Dictionary (R/3 Repository).

Changing a Material BOM

You can use this function to change a material BOM in the SAP System.

Module name:	RFC_CHANGE_BILL_OF_MATERIAL
C prototype:	*int CadRfcBomChange(BICSK *, CadRfcImportField *, ITAB_H **, ITAB_H **)* *int CadRfcWriteBomTabEntry(char*, char*, int)*

First, this function module goes to the initial BOM maintenance screen. All parameters of the initial screen that were transferred with the function call are already filled out. As a security mechanism, it is no longer possible to change the values in the fields *Material number*, *Plant*, *Usage,* and *Alternative* that were transferred via the interface.

The complete BOM must be transferred from the external system. The SAP System automatically compares the items of the old and new BOM. A CAD indicator is set in the item records of all items transferred from the external system. Once the BOMs have been compared and the old BOM has been changed, you see the item overview screen of the BOM. You can change the BOM before you save, if required.

Please also note the following when changing a BOM:

CAD indicator

The comparison only takes in the items that already have a CAD indicator in the old BOM. This prevents you from using the interface program to change items created in the online SAP System.

Overwriting values in identical items

The comparison recognizes an item in the new BOM as identical to an item in the old BOM if the item number and component name are the same. In this case, the parameters transferred from the external system overwrite the old item data. Fields that are transferred with their initial values have not effect on existing item data.

Resetting values

If you want to replace an existing field value with the initial value, transfer the special character '!' for this parameter from the external system.

History

All changes to BOMs can be made with or without history.

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
MATERIAL_HEADER_DATA	Structure	BICSK	BOM header data

Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_MAT_HEADER_DATA	Structure	BICSK	Changed BOM header data
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use
DMS_CLASS_DATA	Structure	CSRFCITEM	BOM item data

Import parameters

MATERIAL_HEADER_DATA

In this structure, you can transfer any BOM header data that can also be maintained online. You can call the precise structure of base structure *BICSK* via the SAP Dictionary (R/3 Repository).

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

UPDATE	BOM changed
Space	Other

EXP_MAT_HEADER_DATA

If a BOM has been changed, the current BOM header data is made available in structure *EXP_MAT_HEADER_DATA*.

Tables

DMS_CLASS_DATA

Table structure *ITEM_DATA* can be used to transfer the item data of the changed BOM. You can call the precise structure of base structure *CSRFCITEM* via the SAP Dictionary (R/3 Repository).

Displaying a Material BOM

You can use this function to display a material BOM in the SAP System.

Module name:	RFC_DISPLAY_BILL_OF_MATERIAL
C prototype:	*int CadRfcBomDisplay(CadRfcExportField *, CadRfcImportField *,ITAB_H **, ITAB_H **)*

The following display functions are supported:

- *Explode BOM level by level* (online transaction CS11 - single-level)
- *Explode BOM level by level* (online transaction CS11 - multi-level)
- *Explode multi-level BOM* (online transaction CS12)
- *Display summarized BOM* (online transaction CS13)
- *Display single-level BOM* (online transaction CS03)

All parameters of the initial screen that were transferred with the function call are entered on the initial screen. There is a parameter for defining whether the initial screen is skipped or not.

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
MATERIAL	CHAR	18	Material number
BOM_ACTION	CHAR	1	Display function.
BOM_APPLICATION	CHAR	4	Application
PLANT	CHAR	4	Plant
BOM_ALTERNATIVE	CHAR	2	Alternative
CHANGE_NUMBER	CHAR	12	Change number
VALID_FROM	CHAR	10	Valid-from date
BOM_TYPE	CHAR	1	BOM usage
REVISION_LEVEL	CHAR	2	Revision level
SKIP_FIRST_SCREEN	CHAR	1	Skip initial screen
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

MATERIAL

Material number of the BOM.

BOM_ACTiON

Display function.

Display functions for BOMs

Value	Description
1	*Explode BOM level by level* (online transaction CS11 - single-level)
2	*Explode BOM level by level* (online transaction CS11 - multi-level)
3	*Explode multi-level BOM* (online transaction CS12)
4	*Display summarized BOM* (online transaction CS13)
5	*Display single-level BOM* (online transaction CS03)

BOM_APPLICATION

BOM application

Required for determining the appropriate alternative in display functions 1-4.

PLANT

Plant for which you want to explode the BOM.

BOM_ALTERNATIVE

Alternative BOM

Can only be entered for display functions 1-4.

CHANGE_NUMBER

Change number

The BOM is displayed on the valid-from date of the change number you enter.

VALID_FROM

Valid-from date

The BOM becomes effective on the valid-from date of the change number you enter.

BOM_TYPE

BOM usage

The usage shows the area within the company for which the BOM is effective. You can only enter this parameter for display function 5.

REVISION_LEVEL

Revision level

SKIP_FIRST_SCREEN

Space

You can select additional selection options by making entries on the initial screen of the display transaction.

X

Unless a warning message or error message is triggered, the initial screen is processed in the background, and you go directly to the exploded BOM.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

Functions for Plant Maintenance

The following functions are supported for plant maintenance:

Equipment master record

Creating an Equipment Master Record [Seite 513]
Changing an Equipment Master Record [Seite 515]
Displaying an Equipment Master Record [Seite 516]
Reading an Equipment Master Record [Seite 518]

Functional location

Creating a Functional Location [Seite 520]
Changing a Functional Location [Seite 522]
Displaying a Functional Location [Seite 524]
Reading a Functional Location [Seite 526]

Maintenance notification

Creating a Maintenance Notification [Seite 528]
Changing a Maintenance Notification [Seite 530]
Displaying a Maintenance Notification [Seite 531]
Creating a Maintenance Order [Seite 532]
Changing a Maintenance Order [Seite 534]
Displaying a Maintenance Order [Seite 535]

Creating an Equipment Master Record

You can use this function to create an equipment master record.

Module name:	RFC_CREATE_EQUIPMENT_MASTER
C prototype:	*int CadRfcEquCreate(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
IMP_EQUIPMENT	CHAR	18	Equipment number
EQUIPMENT_TYPE	CHAR	1	Equipment type
VALID_FROM	CHAR	10	Valid-from date
REF_EQUIPMENT	CHAR	18	Copy from equipment number
REF_MATERIAL	CHAR	18	Copy from material number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_EQUIPMENT	CHAR	18	Equipment number
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

IMP_EQUIPMENT, EQUIPMENT_TYPE

If you enter both an equipment number and an equipment type, the initial screen is processed in the background (provided that no error occurs).

This function is not yet supported in release 3.0A (you cannot transfer an equipment number).

VALID_FROM

Valid-from date of equipment.

The default is the current date.

REF_EQUIPMENT, REF_MATERIAL

Equipment number and material number for copying from an existing record.

Export parameters

EXP_EQUIPMENT

This field contains the assigned equipment number.

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
UPDATE	Equipment master created (not in 3.0A)
Space	Other

Before you can process original application data with function code *DOC_INSERT,* you need to call function module *RFC_INSERT_DRAWING*.

If you link an equipment to a document as soon as you change the equipment, you can branch from the equipment master maintenance function to the document management system. If you request a document to insert, the equipment master maintenance function you started is terminated, and any data you entered will be lost.

Changing an Equipment Master Record

You can use this function to change an equipment master record.

Module name:	RFC_CHANGE_EQUIPMENT_MASTER
C prototype:	*int CadRfcEquChange(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: EQUIPMENT	CHAR	18	Equipment number
Export parameters: FUNCTION_CODE	CHAR	14	Function code
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Not currently in use Not currently in use

Import parameters

EQUIPMENT

If you enter an equipment number, the initial screen is processed in the background (provided that no error occurs).

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
UPDATE	Equipment master changed (not in 3.0A)
Space	Other

Before you can process original application data with function code *DOC_INSERT,* you need to call function module *RFC_INSERT_DRAWING*.

If you link an equipment to a document as soon as you change the equipment, you can branch from the equipment master maintenance function to the document management system. If you request a document to insert, the equipment master maintenance function you started is terminated, and any data you entered will be lost.

Displaying an Equipment Master Record

You can use this function to display an equipment master record.

Module name:	RFC_DISPLAY_EQUIPMENT_MASTER
C prototype:	*int CadRfcEquDisplay(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: EQUIPMENT VALID_FROM	 CHAR CHAR	 18 10	 Equipment number Valid-from date
Export parameters: FUNCTION_CODE	 CHAR	 14	 Function code
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Table with equipment data Not currently in use

Import parameters

EQUIPMENT

If you enter an equipment number, the initial screen is processed in the background (provided that no error occurs).

VALID_FROM

Valid-from date of equipment.

The default is the current date.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

Tables

SAP_FIELD_DATA

All fields for the material master structure MARA and the description (MAKT – MAKTX) are returned in this table.

The fields in table SAP_FIELD_DATA are filled as follows:

FIELDMULTI	Language-dependent description of the database field
FIELDNAME	Field name (for example, EQUI – EQUNR)
FIELDVALUE	Value

You can read an entry in the table SAP_FIELD_DATA with the function CadRfcGetFieldTabEntry.

Reading an Equipment Master Record

You can use this function to read equipment master data.

Module name:	RFC_GET_EQUIPMENT_DATA
C prototype:	*int CadRfcGetEquipmentData (CadRfcExportField *, CadRfcImportField *,ITAB_H **FieldData, ITAB_H **DmsClassData);*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: I_MATERIAL	 CHAR	 18	 Equipment number
Export parameters: E_RETURN E_MESSAGE	 CHAR CHAR	 10 200	 Return code (0 or 99) Error message
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Table with equipment data Not currently in use

Import parameters

I_EQUIPMENT Equipment number

Export parameters

E_RETURN	Return code 0 OK. 99 Error
E_MESSAGE	If there are errors, an error message will be returned in this parameter. In this case, the return code (E_RETURN) is 99.

Tables

SAP_FIELD_DATA

All fields of the equipment structure (EQUI) are returned to this table.

The fields in the SAP_FIELD_DATA table are filled as follows:

FIELDMULTI Language-dependent description of the database field

FIELDNAME Field name (for example, EQUI – EQUNR)

FIELDVALUE Value

You can read an entry in the table SAP_FIELD_DATA with the function CadRfcGetFieldTabEntry.

Creating a Functional Location

You can use this function to create a functional location.

Module name:	RFC_CREATE_FUNC_LOC_MASTER
C prototype:	*int CadRfcFloCreate(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
FUNCTIONAL_LOCATION	CHAR	30	Functional location
STRUCTURE_INDICATOR	CHAR	5	Structure indicator
FUNCTIONAL_LOCATION_TYPE	CHAR	1	Functional location category
REF_FUNCTIONAL_LOCATION	CHAR	30	Copy from functional location Copy from reference functional location
REF_REFERENCE_LOCATION	CHAR	30	
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_FUNCTIONAL_LOCATION	CHAR	18	Functional location
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

IMP_FUNCTIONAL_LOCATION, STRUCTURE_INDICATOR, FUNCTIONAL_LOCATION_TYPE

If you enter a functional location number, a category, and a structure indicator, the initial screen is processed in the background (provided that no error occurs).

REF_FUNCTIONAL_LOCATION

Functional location for copying from (not supported in 3.0A).

REF_REFERENCE_LOCATION

Reference functional location for copying from (not supported in 3.0A)

Export parameters

EXP_ FUNCTIONAL_LOCATION

This field contains the assigned functional location number.

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document

UPDATE	Functional location created (not in 3.0A)
Space	Other

Before you can process original application data with function code *DOC_INSERT,* you need to call function module *RFC_INSERT_DRAWING*.

If you link a functional location to a document as soon as you create the functional location, you can branch from the functional location maintenance function to the document management system. If you request a document to insert, the functional location maintenance function you started is terminated, and any data you entered will be lost.

Changing a Functional Location

You can use this function to change a functional location.

Module name:	RFC_CHANGE_FUNC_LOC_MASTER
C prototype:	*int CadRfcFloChange(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: FUNCTIONAL_LOCATION STRUCTURE_INDICATOR	 CHAR CHAR	 30 5	 Functional location Structure indicator
Export parameters: FUNCTION_CODE	 CHAR	 14	 Function code
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Not currently in use Not currently in use

Import parameters

FUNCTIONAL_LOCATION

If you enter a functional location number, the initial screen is processed in the background (provided that no error occurs).

STRUCTURE_INDICATOR

An optional entry to aid entry.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
UPDATE	Functional location changed (not in 3.0A)
Space	Other

Before you can process original application data with function code *DOC_INSERT,* you need to call function module *RFC_INSERT_DRAWING*.

If you link a functional location to a document as soon as you change the functional location, you can branch from the functional location maintenance function to the document management system. If you request a document to insert, the functional location maintenance function you started is terminated, and any data you entered will be lost.

Displaying a Functional Location

You can use this function to display a functional location.

Module name:	RFC_DISPLAY_FUNC_LOC_MASTER
C prototype:	*int CadRfcFloDisplay(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
FUNCTIONAL_LOCATION	CHAR	30	Functional location
STRUCTURE_INDICATOR	CHAR	5	Structure indicator
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Table with data of functional location
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

FUNCTIONAL_LOCATION

If you enter a functional location number, the initial screen is processed in the background (provided that no error occurs).

STRUCTURE_INDICATOR

An optional entry to aid entry.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

Tables

SAP_FIELD_DATA

All fields for the material master structure MARA and the description (MAKT – MAKTX) are returned in this table.

The fields in table SAP_FIELD_DATA are filled as follows:

FIELDMULTI	Language-dependent description of the database field
FIELDNAME	Field name (for example, IFLO – TPLNR)
FIELDVALUE	Value

You can read an entry in the table SAP_FIELD_DATA with the function CadRfcGetFieldTabEntry.

Reading a Functional Location

You can use this function to read the master data of a functional location.

Module name:	RFC_GET_FUNC_LOC_DATA
C prototype:	*int CadRfcGetFuncLocData (CadRfcExportField *, CadRfcImportField *,ITAB_H **FieldData, ITAB_H **DmsClassData);*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: I_MATERIAL	CHAR	18	Number of the functional location
Export parameters: E_RETURN E_MESSAGE	CHAR CHAR	10 200	Return code (0 or 99) Error message
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	Structure Structure	RFCDMSDATA CLS_CHARAC	Table with data of functional location Not currently in use

Import parameters

I_FUNC_LOC Number of the functional location

Export parameters

E_RETURN	Return code 0 OK. 99 Error
E_MESSAGE	If there are errors, an error message will be returned in this parameter. In this case, the return code (E_RETURN) is 99.

Tables

SAP_FIELD_DATA

All fields for the material master structure MARA and the description (MAKT – MAKTX) are returned in this table.

The fields in table SAP_FIELD_DATA are filled as follows:

FIELDMULTI	Language-dependent description of the database field
FIELDNAME	Field name (for example, IFLO – TPLNR)
FIELDVALUE	Value

You can read an entry in the table SAP_FIELD_DATA with the function CadRfcGetFieldTabEntry.

Creating a Maintenance Notification

You can use this function to create a maintenance notification.

Module name:	RFC_CREATE_PM_REQUEST_MASTER
C prototype:	*int CadRfcPmrCreate(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
NOTIFICATION_TYPE	CHAR	2	Notification type
IMP_EQUIPMENT	CHAR	18	Equipment number
MATERIAL	CHAR	18	Material number
FUNCTIONAL_LOCATION	CHAR	30	Functional location
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
EXP_NOTIFICATION	CHAR	12	Notification number
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

NOTIFICATION_TYPE

If you enter a valid notification type, the initial screen is processed in the background.

EQUIPMENT

Reference object equipment

MATERIAL

Reference object assembly (transfer not possible in 3.0A).

FUNCTIONAL_LOCATION

Reference object functional location

Export parameters

EXP_ NOTIFICATION

This field contains the assigned notification number.

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

UPDATE	Notification created
Space	Other

Changing a Maintenance Notification

You can use this function to change a maintenance notification.

Module name:	RFC_CHANGE_PM_REQUEST_MASTER
C prototype:	*int CadRfcPmrChange(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **) int CadRfcPmrChange(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: NOTIFICATION	CHAR	2	Notification number
Export parameters: FUNCTION_CODE	CHAR	14	Function code
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	Structure Structure	RFCDMSDATA CLS_CHARAC	Not currently in use Not currently in use

Import parameters

NOTIFICATION

If you enter a valid notification number, the initial screen is processed in the background.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

UPDATE	Notification changed (not in 3.0A)
Space	Other

Displaying a Maintenance Notification

You can use this function to display a maintenance notification.

Module name:	RFC_DISPLAY_PM_REQUEST_MASTER
C prototype:	*int CadRfcPmrDisplay(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
NOTIFICATION	CHAR	2	Notification number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

NOTIFICATION

If you enter a valid notification number, the initial screen is processed in the background.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE always returns the value SPACE.

Creating a Maintenance Order

You can use this function to create a maintenance order.

This function is not yet supported in release 3.0.

Module name:	RFC_CREATE_PM_ORDER
C prototype:	*int CadRfcPmOrderrCreate(CadRfcExportField *,CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
PM_TYPE	CHAR	2	PM order type
EQUIPMENT	CHAR	18	Equipment number
MATERIAL	CHAR	18	Material number
FUNCTIONAL_LOCATION	CHAR	30	Functional location
PLANT	CHAR	4	Planning plant
BUSINESS_AREA	CHAR	4	Business area
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
PM_ORDER_NUMBER	CHAR	12	PM order number
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

PM_TYPE

Order type

EQUIPMENT, MATERIAL, FUNCTIONAL_LOCATION

Reference objects: equipment, assembly, or functional location

PLANT, BUSINESS_AREA

Maintenance planning plant and business area

Export parameters

PM_ORDER_NUMBER

This field contains the assigned order number.

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

UPDATE	Order created
Space	Other

Changing a Maintenance Order

You can use this function to change a maintanance order.

This function is not yet supported in release 3.0.

Module name:	RFC_CHANGE_PM_ORDER
C prototype:	*int CadRfcPmOrderrChange(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
PM_ORDER_NUMBER	CHAR	2	PM order number
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

PM_ORDER_NUMBER

If you enter a valid order number, the initial screen is processed in the background.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

UPDATE	Order changed (not in 3.0A)
Space	Other

Displaying a Maintenance Order

You can use this function to display a maintanance order.

This function is not yet supported in release 3.0.

Module name:	RFC_DISPLAY_PM_ORDER
C prototype:	*int CadRfcPmOrderrDisplay(CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
PM_ORDER_NUMBER	CHAR	2	PM order type
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

PM_ORDER_NUMBER

If you enter a valid order number, the initial screen is processed in the background.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE always returns the value SPACE.

Mail and Workflow Functions

The following mail and workflow functions are supported:

Displaying a Mail Inbox [Seite 537]

Displaying a Mail Outbox [Seite 538]

Displaying a Workflow Inbox [Seite 539]

Displaying a Mail Inbox

You can use this function to display a mail inbox.

Module name:	RFC_DISPLAY_MAIL_INBOX
C prototype:	*int CadRfcMailInbox(CadRfcExportField *, CadRfcImportField *,ITAB_H **,ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: None			
Export parameters: FUNCTION_CODE	CHAR	14	Function code
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Not currently in use Not currently in use

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

Displaying a Mail Outbox

You can use this function to display a mail outbox.

Module name:	RFC_DISPLAY_MAIL_OUTBOX
C prototype:	*int CadRfcMailOutbox(CadRfcExportField *, CadRfcImportField*,ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: None			
Export parameters: FUNCTION_CODE	CHAR	14	Function code
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Not currently in use Not currently in use

Export parameters

FUNCTION_CODE

Not currently in use

Displaying a Workflow Inbox

You can use this function to display a workflow inbox.

Module name:	RFC_DISPLAY_WORKFLOW_INBOX
C prototype:	*int CadRfcWorkflowInbox(CadRfcExportField *, CadRfcImportField*,ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: None			
Export parameters: FUNCTION_CODE	CHAR	14	Function code
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Not currently in use Not currently in use

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

General Functions in the SAP Environment

Displaying a Product Structure [Seite 541]
Displaying a Where-Used List for a Material or Document [Seite 543]
Finding Objects via Matchcode [Seite 545]
Finding Objects in Classes [Seite 547]
Requesting SAP System Parameters [Seite 550]

Displaying a Product Structure

This function allows you to display the structure overview for a material or a document by entering the material number or document key, respectively.

Module name:	RFC_EXPLODE_PRODUCT_STRUCTURE
C prototype:	*int CadRfcProductStructure(CadRfcExportField *,CadRfcImportField *, ITAB_H **, ITAB_H **)*

The object you start this function with can be either a material or a document. If you enter both, the material overrides the document. At present, you cannot enter a change number to start this function.

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
MATERIAL	CHAR	18	Material number
DOC_NUMBER	CHAR	25	Document number
DOC_TYPE	CHAR	3	Document type
DOC_PART	CHAR	2	Document part
DOC_VERS	CHAR	2	Document version
BOM_APPLICATION	CHAR	4	Application
PLANT	CHAR	4	Plant
BOM_ALTERNATIVE	CHAR	2	Alternative
CHANGE_NUMBER VALID_FROM	CHAR	12	Change number
BOM_TYPE	CHAR	10	Valid-from date
	CHAR	1	BOM usage
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

MATERIAL

Initial material

DOC_NUMBER, DOC_TYPE, DOC_PART, DOC_VERS

Document key

You can only display the product structure if you make entries in all four key fields.

BOM_APPLICATION, BOM_ALTERNATIVE, BOM_TYPE, PLANT, VALID_FROM

If you select a material to start this function, you can enter these parameters for more detail. These parameters are ignored if you start the function with a document.

CHANGE_NUMBER

Not currently supported.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING*.

Displaying a Where-Used List for a Material or Document

You can use this function to display a where-used list for a material or document by entering the material number or document key, respectively.

Module name:	RFC_DISPLAY_OBJECT_WHERE_USED
C prototype:	*int CadRfcBomWhereUsed(CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*

The object you start this function with can be either a material or a document. If you enter both, the material overrides the document.

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
MATERIAL	CHAR	18	Material number
DOC_NUMBER	CHAR	25	Document number
DOC_TYPE	CHAR	3	Document type
DOC_PART	CHAR	2	Document part
DOC_VERS	CHAR	2	Document version
BOM_APPLICATION	CHAR	4	Application
PLANT	CHAR	4	Plant
BOM_ALTERNATIVE	CHAR	2	Alternative
CHANGE_NUMBER VALID_FROM	CHAR	12	Change number
VALID_TO	CHAR	10	Valid-from date
SKIP_FIRST_SCREEN	CHAR	10	Valid-to date
	CHAR	1	Skip initial screen
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	Not currently in use
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use

Import parameters

MATERIAL

Initial material

DOC_NUMBER, DOC_TYPE, DOC_PART, DOC_VERS

Document key fields

VALID_FROM, VALID_TO

Effectivity period for where-used list

If you do not make an entry here, the current date is entered automatically.

PLANT

Plant

If you enter special character '*', you can display a where-used list that covers all plants.

SKIP_FIRST_SCREEN

Space

You can select additional selection options by making entries on the initial screen of the display transaction.

X

Unless a warning message or error message is triggered, the initial screen is processed in the background, and you go directly to the where-used list.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING*.

Finding Objects via Matchcode

You can use this function to call the SAP matchcode functionality.

Matchcodes for the following objects are supported:

- Material master record
- Document info record
- Change master record
- Equipment master record
- Functional location
- Maintenance notification
- Maintenance order

Module name:	RFC_SELECT_OBJ_VIA_MATCHCODE
C prototype:	*int CadRfcMatchcode(CadRfcExportField *, CadRfcImportField *,ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters: OBJECT_ID RETURN_MCD_VALUES	 CHAR CHAR	 1 1	 Matchcode ID Flag: return selected object
Export parameters: FUNCTION_CODE	 CHAR	 14	 Function code
Tables: SAP_FIELD_DATA DMS_CLASS_DATA	 Structure Structure	 RFCDMSDATA CLS_CHARAC	 Key of selected object Not currently in use

Import parameters

OBJECT_ID

Defines which matchcode object you want to find.

The following values are possible:

M	Material master record
D	Document info record
C	Change master record
E	Equipment master record
F	Functional location
P	Maintenance notification
O	Maintenance order

RETURN_MCD_VALUES

This indicator determines whether the system displays the object that you select (“ “), or whether the object is relayed directly to the external system (“X”). The object key can be determined from the table SAP_FIELD_DATA.

Export parameters

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO OBJ_SELECT	Document changed and checked out Object has been selected
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

If you have selected an object and not started any other action (for example, editing a document), then the transfer parameter FUNCTION_CODE is assigned the value OBJ_SELECT.

SAP_FIELD_DATA

This table is filled with the key of the selected object, irrespective of whether RETURN_MCD_VALUES has been set or not.

You can read the values from the table with the function CadRfcGetFieldTabEntry. The name of the key field is copied into the FieldName attribute. The object key is in the FieldValue attribute.

The following descriptions are used for the key fields:

Material number	MATERIAL
Equipment number	EQUIPMENT
Functional location	FUNC_LOCATION
Malfunction report	NOTIFICATION
PM order	PM_ORDER
Change number	CHANGE_NO
Document key	DOC_NUMBER
	DOC_TYPE
	DOC_PART
	DOC_VERS

Finding Objects in Classes

You can use this function to call the SAP function for finding objects in a class. This function is supported for all classifiable objects.

Examples of the object types you can find:

- Material master record
- Document info record
- Change master record
- Equipment master record
- Functional location

Module name:	RFC_SELECT_OBJECTS_VIA_CLASS
C prototype:	*int CadRfcClassSelection(CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **, ITAB_H **)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Import parameters:			
CLASS_NO	CHAR	18	Class name Class type
CLASS_TYPE	CHAR	3	Valid-from date
VALID_FROM	CHAR	10	
Export parameters:			
FUNCTION_CODE	CHAR	14	Function code
CLASS_NO	CHAR	18	Class number
CLASS_TYPE	CHAR	3	Class type
OBJECT_TABLE	CHAR	10	Name of data base table
OBJECT_KEY	CHAR	50	Object key
VALID_FROM	CHAR	10	Valid-from date
Tables:			
SAP_FIELD_DATA	Structure	RFCDMSDATA	1 - table of selected objects (>1)
DMS_CLASS_DATA	Structure	CLS_CHARAC	Not currently in use
CLASS_DATA	Structure	CLS_CHARAC	Table with assigned characteristic values

You have the option of calling this function module with a class name and class type. If you do **not** enter a class, you can find a class afterward on the initial screen using, for example, matchcode selection or the class hierarchy.

If you selected a document to insert or if a document was checked out during the find objects function, the SAP System makes the relevant class data and assigned characteristic values available to the program that called the function.

You can copy objects from the object list to the external application by using the *Copy* function.

Import parameters

CLASS_NO

Class name

CLASS_TYPE

The class type is used to define the object type you want to find.

VALID_FROM

You can enter a valid-from date for some object types in order to do a time-dependent search.

Export parameters

CLASS_NO

If you selected a document to insert or if a document was checked out, the class name used for the find objects function is entered in this field.

The selected class need not be a document class. For example, you also have the option of starting the function with a material class. You can find a material first, then find the document that is linked to the selected material. You can then insert or check out the document. The SAP System then transfers information on the material class and the characteristic values assigned to the material to the program that called the function.

The class name is also transferred if exactly one object was selected and transferred.

CLASS_TYPE

Class type of the class.

OBJECT_TABLE

Name of the database table for which the class type is defined:

Material master record: MARA

Document master record: DRAW

Equipment master record: EQUI

Functional location: IFLOT

Change master record: AENR

This function can return other objects that are not described here. You must know the structure of the object key for these objects.

OBJECT_KEY

Object key of the selected object

Material master record: 18-character material number

Document info record: 25-character document number, 3-character document type, 3-character document part, and 2-character document version

Equipment master record: 18-character equipment number

Functional location: 30-character functional location number

Change master record: 12-character change number

VALID_FROM

Valid-from date entered as selection criterion.

FUNCTION_CODE

Transfer parameter FUNCTION_CODE can have the following values:

DOC_INSERT	Request to insert a document
DOC_CO	Document changed and checked out
Space	Other

Before you can process original application data with function code *DOC_INSERT* and *DOC_CO,* you need to call function module *RFC_INSERT_DRAWING.*

Tables

CLASS_DATA

If a document was selected to be inserted or checked out, the characteristic values assigned to the object found with the above class shown is shown here. This is also the case if exactly one object from the object list was chosen to be transferred.

Table structure

Description	Format	Length
Characteristic name	CHAR	30
Characteristic description	CHAR	30
Characteristic value	CHAR	30
DIN code letter	CHAR	3
Unit of measure	CHAR	6

SAP_FIELD_DATA

All the objects that have been chosen to be transferred are displayed in this table:

Table structure

Description	Format	Length
FieldMulti	CHAR	40 (not in use)
Data base name	CHAR	35
Object key	CHAR	255

Requesting SAP System Parameters

You can use this function to request the following SAP System parameters:

- User name
- Logon language
- Date format
- Decimal point
- SAP release

Module name:	RFC_GET_SAP_SYSTEM_PARAMETERS
C prototype:	*int CadRfcGetSystemParas(CadRfcImportField *)*

Possible Transfer Parameters

Table of possible transfer parameters

RFC Parameter name	Format	Length	Description
Export parameters:			
DATE_FORMAT	CHAR	30	date format that is set
DECIMAL_SIGN	CHAR	1	decimal point that is set
SAP_SYSTEM_RELEASE	CHAR	4	SAP release
USER_NAME	CHAR	12	User name
LANGUAGE	CHAR	1	Logon language

Description of Dialog RFC Function Library cadrfc00

This section is a detailed description of the call syntax for each function call in the CAD function library.

You can use example program *cadrftst*, also supplied as a C source file with the workstation software, to test the interface. You can also use this program as a basis for your own developments. You can establish a connection to the external system from your R/3 System. As of release 3.0C, you can also establish a connection directly from the example program (call: cadrftst/r3connect).

Description of Dialog RFC Function Library cadrfc00

The Dialog RFC Function Library [Seite 553]

Header Files and Libraries Required [Seite 554]

Header File cadrfc.h [Seite 555]

Table Structures [Seite 557]

General functions [Seite 558]

Establishing an RFC Connection (Starting from External System) [Seite 559]

Terminating an RFC Connection [Seite 561]

Initializing Transfer Structures [Seite 562]

Querying a Communication Handle [Seite 563]

Defining a Communication Handle [Seite 564]

Return of Error Messages [Seite 565]

Functions for Document Management [Seite 571]

Functions for Material Master Records [Seite 566]

Functions for Engineering Change Management [Seite 584]

Mail and Workflow Functions [Seite 603]

BOM Functions [Seite 578]

General Functions in the SAP Environment [Seite 607]

Displaying a Product Structure [Seite 608]

Requesting SAP System Parameters [Seite 609]

Displaying a Where-Used List for a Material or Document [Seite 610]

Finding Objects via Matchcode [Seite 611]

Finding Objects in Classes [Seite 617]

Functions for Plant Maintenance [Seite 588]

Using Calls in the SAP-CAD Function Library [Seite 620]

The Dialog RFC Function Library

The CAD function library simplifies dialog RFC access, as an additional option to calling SAP function modules using the standard SAP RFC syntax.

For example, the CAD function library has the following advantages:

- The function modules already contain the parameters required.
- You can store default values in file *cadrfc.ini.* These values are set automatically before each function call.

For most calls in the CAD function library, there is a corresponding RFC function module.

See also:

Description of the Available RFC Function Modules [Seite 471]

Header Files and Libraries Required

Before you can work with the dialog RFC function library, the header file cadrfc.h [Seite 555] must be installed in the application program. This header file then accesses the standard RFC header files *saprfc.h* and *sapitab.h*.

When you link the application, you must install the dialog RFC library *cadrfc00.lib* or *cadrfc00.o*. If you also want SAP table objects to be accessed, you must also install the standard RFC library *librfcxx.lib* (xx = 16 or 32) or *librfc.o*.

Header File cadrfc.h

This file contains the function prototypes and the definitions of transfer structures.

See also:

File cadrfc.h [Seite 656]

The structures defined are described below.

- **SAP_FIELD_DATA – general data**

  A pointer to a table with this structure is transferred in almost all function calls, but is not currently in use.

- **CLASS_CHARACTERISTICS – structure with data on assigned characteristic values**

  A pointer to a table with this structure is transferred in almost all function calls, but is not currently in use.

  The only exception is the function call: *Find objects in classes* (CadRfcClassSelection).

  In this function call, a pointer to a table of the characteristic values assigned to each object found is made available.

- **BICSK – BOM header data**

  This structure is used to define the possible transfer fields for the BOM header.

  The structure is used in the following functions:

  – *Create BOM (CadRfcBomCreate)*
  – *Change BOM (CadRfcBomChange)*
  – *Create BOM with sub-items (CadBomCreateSubItem)*
  – *Read and change BOM with sub-items (CadBomChangeSubItem)*
  – *Change BOM with sub-items (CadBomChangeSubItemRead)*
  – *Display BOM with sub-items (CadBomDisplaySubItem)*

  This structure also exists in the R/3 System with the same structure and the same name. For this reason, information on the individual fields can be called from the R/3 Repository.

- **CSRFCITEM – BOM item data**

  This structure is used to define the possible transfer fields for the BOM items.

  The structure is used in the following functions:

  – *Create BOM (CadRfcBomCreate)*
  – *Change BOM (CadRfcBomChange)*
  – *Create BOM with sub-items (CadBomCreateSubItem)*
  – *Read and change BOM with sub-items (CadBomChangeSubItem)*
  – *Change BOM with sub-items (CadBomChangeSubItemRead)*

- *Display BOM with sub-items (CadBomDisplaySubItem)*

This structure also exists in the R/3 System with the same structure and the same name. For this reason, information on the individual fields can be called from the R/3 Repository.

- **CSSUBITEM – BOM sub-item data**

This structure is used to define the possible transfer fields for the BOM sub-items.

The structure is used in the following functions:

- *Create BOM with sub-items (CadBomCreateSubItem)*
- *Read and change BOM with sub-items (CadBomChangeSubItem)*
- *Change BOM with sub-items (CadBomReadSubItem)*
- *Display BOM with sub-items (CadBomDisplaySubItem)*

This structure also exists in the R/3 System with the same structure and the same name. For this reason, information on the individual fields can be called from the R/3 Repository.

- **DRAW_RFC – document info record data**

This structure is used to define the possible transfer fields for maintaining document info records.

The structure is used in the following functions:

- *Create document info record (CadRfcDocCreate)*
- *Change document info record (CadRfcDocChange)*
- *Insert document info record (CadRfcDocInsert)*
- *Read document info record (CadRfcGetDmsData)*

This structure also exists in the R/3 System with the same structure and the same name. For this reason, information on the individual fields can be called from the R/3 Repository.

- **CadRfcExportField – general export data**

This structure is used to define general transfer fields to be sent from an external application to the SAP System. The structure is used in almost all function calls.

- **CadRfcImportField – general import data**

This structure is used to define general transfer fields to be sent from the SAP System to the external application. The structure is used in almost all function calls.

- **CadRfcInitData – initialization data**

Table Structures

In a number of function calls, two pointers of type ITAB_H must be transferred. The pointers refer to SAP table structures that are not currently in use. For a function call, you only need the simple definition of these pointers.

Example:

```
#include "cadrfc.h"
ITAB_H        *FldDataPtr;
ITAB_H        *DmsCharacPtr;
RFC_RC         CadRfcRc;
CadRfcExportField *RfcExport;
CadRfcImportField *RfcImport;

CadRfcRc = CadRfcXY(RfcExport, RfcImport, &FldDataPtr, &DmsCharacPtr);
```

General Functions

The following general functions are supported:

Establishing an RFC Connection (Starting from External System) [Seite 559]
Terminating an RFC Connection [Seite 561]
Initializing Transfer Structures [Seite 562]
Querying a Communication Handle [Seite 563]
Defining a Communication Handle [Seite 564]

Establishing an RFC Connection (Starting from External System)

This function is used when you start the RFC connection to an R/3 System from the external application.

C prototype:	int CadRfcOpen (char*, char*, char*, char*, char*, char*, char*, RFC_HANDLE*)	
RFC module:	RfcOpen	
Import parameters:	char* Destination (log. name of R/3 System) char* Hostname (application server) char* Sysnr (R/3 System ID) char* User (user) char* Password (password) char* Client (client) char* Language (logon language)	
Export parameters:	Communication handle	
Return parameters:	0 215 216 217	o.k. File *cadrfc.ini* not found Entry not found in cadrfc.ini Error calling *RfcOpen*

All transfer fields are optional. If you do not make an entry in a field, the program tries to read default values from configuration *cadrfc.ini*.

You have two options for establishing a connection:

- **Enter a *Destination***

  You can enter a logical name for an R/3 System for parameter *Destination*. The communication parameters for this R/3 System can either be specified in the CADRTC.INI file, or there must be a corresponding entry in the SAPRFC.INI file. This entry is used to find the R/3 System you require.

- **Enter *Hostname* and *Sysnr***

  You can enter the application server and system ID of the R/3 System you require directly. Configuration file *saprfc.ini* is not used in this case.

  The values for *Hostname* and *Sysnr* have priority when you enter the three parameters. The contents of the *Destination* parameter would be ignored in this case.

You can enter the parameters required for logging onto the R/3 System in the fields *CadRfcUser, CadRfcPassword, CadRfcClient,* and *CadRfcLanguage*.

Establishing an RFC Connection (Starting from External System)

If an error occurs in this function, you automatically see a dialog box containing an error message on Windows platforms. On Unix, the error is just logged to file dev_rfc.

Terminating an RFC Connection

This function is used to terminate the RFC connection between the external application and the R/3 System.

C prototype:	*int CadRfcClose* ()	
RFC module:	RfcClose	
Return parameters:	0 204	o.k. No connection currently exists.

Initializing Transfer Structures

This function allocates memory to transfer structures or tables and returns a pointer to the memory areas.

C prototype:	*int CadRfcAllocMemory* (CadRfcExportField **, CadRfcImportField **, BICSK **, DRAW_RFC **)	
RFC module:	-	
Export parameters:	Pointer to memory area for CadRfcExportField Pointer to memory area for CadRfcImportField Pointer to memory area for BICSK Pointer to memory area for DRAW_RFC	
Return parameters:	0 205-211	o.k. Error allocating memory

This function must be called before you call a function that accesses one of these pointers for the first time.

Querying a Communication Handle

You can use this function to query a set RFC communication handle.

C prototype:	*int CadRfcGetHandle* (RFC_HANDLE*)	
RFC module:	-	
Export parameters:	RFC communication handle	
Return parameters:	0 204	o.k. No connection currently exists.

Defining a Communication Handle

You can use this function to transfer an external RFC communication handle to the CAD RFC function library.

C prototype:	*int CadRfcSetHandle* (RFC_HANDLE*)	
RFC module:	-	
Import parameters:	RFC communication handle	
Return parameters:	0	o.k.

Any existing communication handle is overwritten. All RFC calls as of this moment use the new communication handle.

Return of Error Messages

Up until release 4.0A, it has not been possible to forward error texts from dialog RFC functions to the external application. You can now do this with release 4.0 because the following two new fields have been added to the structure *CadRfcImportField*:

- *Message* (message text)
- *MessageLength* (length of the message)

You can attach values to these parameters after you have called a dialog RFC function.

You can find an example of this in CADRFTST.C.

Functions for Material Master Records

The following functions are supported for material master records:

Creating a Material Master Record [Seite 567]
Changing a Material Master Record [Seite 568]
Displaying a Material Master Record [Seite 569]
Reading a Material Master Record [Seite 570]

Creating a Material Master Record

You can use this function to create a material master record.

C prototype:	*int CadRfcMatCreate (CadRfcExportField *, CadRfcImportField *, ITAB_H **,ITAB_H **)*	
RFC module:	RFC_CREATE_MATERIAL_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code) CadRfcImportField->Material (allocated material number)	
Export parameters:	CadRfcExportField->Material (material number) CadRfcExportField->MatApplication (material category)) CadRfcExportField->MatType (material type) CadRfcExportField->RefMaterial (copy from material number)	
Return parameters:	0 1-9	o.k. RFC communication error

The *Tcode* field in the BICSK structure has been lengthened from 4 to 20 bytes between release 3.0 and release 4.0. To ensure the downward compatability of the CAD interface, access for external applications will still be via the 3.0 structure. When accessing a 4.0 system, the CAD interface will ensure that the structure sent reaches SAP in the correct format. When a BICSK structure is received, you should note that, on access to a 4.0 system, the *Tcode2* attribute is now filled instead of the *Tcode* attribute.

Changing a Material Master Record

You can use this function to change a material master record.

C prototype:	*int CadRfcMatChange (CadRfcExportField *,CadRfcImportField *, ITAB_H **,ITAB_H **)*	
RFC module:	RFC_CHANGE_MATERIAL_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcImportField->FunctionCode (Function code) CadRfcExportField->ChangeNumber (change number)	
Return parameters:	0	o.k.

The *Tcode* field in the BICSK structure has been lengthened from 4 to 20 bytes between release 3.0 and release 4.0. To ensure the downward compatability of the CAD interface, access for external applications will still be via the 3.0 structure. When accessing a 4.0 system, the CAD interface will ensure that the structure sent reaches SAP in the correct format. When a BICSK structure is received, you should note that, on access to a 4.0 system, the *Tcode2* attribute is now filled instead of the *Tcode* attribute.

Displaying a Material Master Record

You can use this function to display a material master record.

C prototype:	*int CadRfcMatDisplay (CadRfcExportField *, CadRfcImportField*, ITAB_H **,ITAB_H **)*	
RFC module:	RFC_DISPLAY_MATERIAL_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->Material (material number)	
Return parameters:	0 1-9	o.k. RFC communication error

The *Tcode* field in the BICSK structure has been lengthened from 4 to 20 bytes between release 3.0 and release 4.0. To ensure the downward compatability of the CAD interface, access for external applications will still be via the 3.0 structure. When accessing a 4.0 system, the CAD interface will ensure that the structure sent reaches SAP in the correct format. When a BICSK structure is received, you should note that, on access to a 4.0 system, the *Tcode2* attribute is now filled instead of the *Tcode* attribute.

Reading a Material Master Record

You can read material master data with this function. The MARA structure and the short text MAKT fields are returned to the *FieldData* table.

You can read an entry by using the *CadRfcGetFieldTabEntry*. The fields in the *FieldData* table are filled as follows:

- *Fieldmulti* (language-dependent short text of the data base field)
- *Fieldname* (field name, for example: MARA-MATNR)
- *Fieldvalue* (value)

C prototype:	*int CadRfcGetMaterialData (CadRfcExportField *ExportData, CadRfcImportField *ImportData, ITAB_H *FieldData, ITAB_H *DmsClassData);*	
RFC module:	RFC_GET_MATERIALMASTER	
Import parameters:	ITAB_H FieldData ImportData -> Message ImportData -> MessageLength	
Export parameters:	ExportField -> Material (material number)	
Return parameters:	0 1-9 99	o.k. RFC communication error Application errorr

Functions for Document Management

The following document management records are supported:

Creating a Document Info Record [Seite 572]
Changing a Document Info Record [Seite 573]
Displaying a Document Info Record [Seite 574]
Finding a Document Info Record [Seite 575]
Inserting a Document Info Record [Seite 576]
Reading a Document Info Record [Seite 577]

Creating a Document Info Record

You can use this function to create a document info record.

C prototype:	int CadRfcDocCreate (DRAW_RFC *, CadRfcImportField *, ITAB_H **, ITAB_H **);	
RFC module:	RFC_CREATE_DOCUMENT_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code) DRAW_RFC-Structure (document data)	
Export parameters:	DRAW_RFC-Structure (document data)	
Return parameters:	0 1-9	o.k. RFC communication error

Changing a Document Info Record

You can use this function to change a document info record

C prototype:	int CadRfcDocChange (DRAW_RFC *, CadRfcImportField *, ITAB_H **, ITAB_H **);	
RFC module:	RFC_CHANGE_DOCUMENT_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code) DRAW_RFC-Structure (document data)	
Export parameters:	DRAW_RFC-Structure (document data)	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying a Document Info Record

You can use this function to display a document info record.

C prototype:	*int CadRfcDocDisplay (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_DOCUMENT_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->DocNumber (document number) CadRfcExportField->Dokar (document type) CadRfcExportField->Doktl (document part) CadRfcExportField->Dokvr (document version)	
Return parameters:	0 1-9	o.k. RFC communication error

Finding a Document Info Record

You can use this function to find a document info record.

C prototype:	*int CadRfcDocSelect (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_SELECT_DOCUMENT_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->DocNumber (document number) CadRfcExportField->Dokar (document type) CadRfcExportField->Doktl (document part) CadRfcExportField->Dokvr (document version) CadRfcExportField->Dwnam (user name) CadRfcExportField->ChangeNumber (change number) CadRfcExportField->Labor (laboratory/design office) CadRfcExportField->Dokst (document status) CadRfcExportField->SkipFirstScreen (skip initial screen)	
Return parameters:	0 1-9	o.k. RFC communication error

Inserting a Document Info Record

You can use this function to make the document key and the access path to the original application file available for a document you want to insert or a document that is checked out.

C prototype:	*int CadRfcDocInsert (DRAW_RFC *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_INSERT_DRAWING	
Import parameters:	CadRfcImportField->FunctionCode (function code) DRAW_RFC-Structure (document data)	
Export parameters:	-	
Return parameters:	0 1-9	o.k. RFC communication error

Reading a Document Info Record

You can use this function to read a document info record.

C prototype:	*int CadRfcGetDmsData (CadRfcExportField *, DRAW_RFC *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_GET_DMS_DATA	
Import parameters:	CadRfcImportField->FunctionCode (function code) DRAW_RFC-Struktur (read document data)	
Export parameters:	CadRfcExportField->DocNumber (document number) CadRfcExportField->Dokar (document type) CadRfcExportField->Doktl (document part) CadRfcExportField->Dokvr (document version)	
Return parameters:	0 1-9	o.k. RFC communication error

Functions for BOMs

The following functions are supported for BOMs:

Creating a Material BOM [Seite 579]
Changing a Material BOM [Seite 580]
Displaying a Material BOM [Seite 581]
Filling out the BOM Item Table [Seite 582]
Reading the BOM Item Table [Seite 583]

Creating a Material BOM

You can use this function to create a material BOM in the SAP System.

C prototype:	*int CadRfcBomCreate (BICSK *, CadRfcImportField *, ITAB_H**, ITAB_H **)*	
RFC module:	RFC_CREATE_BILL_OF_MATERIAL	
Import parameters:	CadRfcImportField->FunctionCode (function code) BICSK structure (BOM header data)	
Export parameters:	BICSK structure (BOM header data)	
Return parameters:	0 1-9	o.k. RFC communication error

Changing a Material BOM

You can use this function to change a material BOM in the SAP System.

C prototype:	int CadRfcBomChange (BICSK *, CadRfcImportField *, ITAB_H**, ITAB_H **)	
RFC module:	RFC_CHANGE_BILL_OF_MATERIAL	
Import parameters:	CadRfcImportField->FunctionCode (function code) BICSK structure (BOM header data)	
Export parameters:	BICSK structure (BOM header data)	
Return parameters:	0 1-9	o.k. RFC communication error

All BOM header data is transferred via the BICSK structure defined in the header file. The BOM header data must be filled out first by calling *CadRfcWriteBomTabEntry* as many times as required.

Displaying a Material BOM

You can use this function to display a material BOM in the SAP System.

C prototype:	*int CadRfcBomDisplay (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_BILL_OF_MATERIAL	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->Material (material number) CadRfcExportField->BomAction (display function) CadRfcExportField->BomApplication (application) CadRfcExportField->Plant (plant) CadRfcExportField->BomAlternative (alternative BOM) CadRfcExportField->ChangeNumber (change number) CadRfcExportField->ValidFrom (valid-from date) CadRfcExportField->BomType (BOM usage) CadRfcExportField->RevisionLevel (revision level) CadRfcExportField->SkipFirstScreen (skip initial screen)	
Return parameters:	0 1-9	o.k. RFC communication error

Filling out the BOM Item Table

This function is used to fill out a table with BOM items.

A line of the table of the BOM item record is filled out by repeating the function call for this function. Each time the table is called, you can transfer an attribute value for an attribute name. The index transferred defines the table line.

The first time you call this function with index 1, the entire table is initialized.

The attribute names match the names of the *CSRFCITEM* structure defined in header file *cadrfc.h.*

The index must be counted up sequentially. If the index changes, the system assumes that the current line is filled out and automatically inserts the line in the table. You can then no longer change the line. If you try to access the line later using a lower index, you see an error message.

To close the function for filling out the BOM item table, use a function call with index 0.

C prototype:	*int CadRfcWriteBomTabEntry (char*, char*, int)*	
RFC module:	-	
Import parameters:	-	
Export parameters:	char* Attributname char* AttributValue int EntryNumber	
Return parameters:	0 212-214	o.k. Table processing errors

Reading the BOM Item Table

This function returns individual BOM items to the ItemData structure. The item to be read is specified in the internal table with the EntryNumber parameter.

C prototype:	*int CadRfcGetBomItemEntry (int EntryNumber, CSRFCITEM *ItemData)*	
RFC module:	-	
Import parameters:	CSRFCITEM * Item Data (an item record)	
Export parameters:	int EntryNumber (number of the entry in the item table)	
Return parameters:	0 218-221	o.k. internal errors

You can find an example of how to use this function in the CADRFTST.C file.

Functions for Engineering Change Management

The following functions for change master records are supported:

Creating a Change Master Record [Seite 585]

Changing a Change Master Record [Seite 586]

Displaying a Change Master Record [Seite 587]

Creating a Change Master Record

You can use this function to create a change master record.

C prototype:	*int CadRfcChgCreate (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_CREATE_CHANGE_MASTER	
Import parameters:	CadRfcExportField->Material (Material number) CadRfcImportField->ChangeNumber (assigned change number)	
Export parameters:	CadRfcExportField->ChangeNumber (change number)	
Return parameters:	0 1-9	o.k. RFC communication error

Changing a Change Master Record

You can use this function to change a change master record.

C prototype:	*int CadRfcChgChange (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_CHANGE_CHANGE_MASTER	
Import parameters:	CadRfcExportField->Material (Material number)	
Export parameters:	CadRfcExportField->ChangeNumber (change number)	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying a Change Master Record

You can use this function to display a change master record.

C prototype:	*int CadRfcChgDisplay (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_CHANGE_ MASTER	
Import parameters:	CadRfcExportField->Material (Material number)	
Export parameters:	CadRfcExportField->ChangeNumber (change number)	
Return parameters:	0 1-9	o.k. RFC communication error

Functions for Plant Maintenance

Equipment master record

Creating an Equipment Master Record [Seite 589]
Changing an Equipment Master Record [Seite 590]
Displaying an Equipment Master Record [Seite 591]
Reading an Equipment Master Record [Seite 592]

Functional location

Creating a Functional Location [Seite 593]
Changing a Functional Location [Seite 594]
Displaying a Functional Location [Seite 595]
Reading an Equipment Master Record [Seite 592]

Maintenance notification

Creating a Maintenance Notification [Seite 597]
Changing a Maintenance Notification [Seite 598]
Displaying a Maintenance Notification [Seite 599]
Creating a Maintenance Order [Seite 600]
Changing a Maintenance Order [Seite 601]
Displaying a Maintenance Order [Seite 602]

Creating an Equipment Master Record

You can use this function to create an equipment master record.

C prototype:	*int CadRfcEquCreate (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_CREATE_EQUIPMENT_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code) CadRfcImportField->Equipment (assigned equipment number)	
Export parameters:	CadRfcExportField->Equipment (equipment number) CadRfcExportField->ValidFrom (valid-from date) CadRfcExportField->EquipmentType (equipment type) CadRfcExportField->RefMaterial (copy from material number) CadRfcExportField->RefEquipment (copy from equipment number)	
Return parameters:	0 1-9	o.k. RFC communication error

Changing an Equipment Master Record

You can use this function to change an equipment master record.

C prototype:	*int CadRfcEquChange (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_CHANGE_EQUIPMENT_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->Equipment (equipment number)	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying an Equipment Master Record

You can use this function to display an equipment master record.

C prototype:	*int CadRfcEquDisplay (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_EQUIPMENT_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->Equipment (equipment number) CadRfcExportField->ValidFrom (valid-from date)	
Return parameters:	0 1-9	o.k. RFC communication error

Reading an Equipment Master Record

You can use this function to read equipment master data. The fields of the EQUI-structure are returned in the *FieldData* table.

You can read an entry by using the *CadRfcGetFieldTabEntry*. The fields in the *FieldData* table are filled as follows:

- *Fieldmulti* (language-dependent short text of the data base field)
- *Fieldname* (field name, for example: MARA-MATNR)
- *Fieldvalue* (value)

C prototype:	*int CadRfcEquDisplay (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_GET_EQUIPMENT_DATA	
Import parameters:	ITAB_H FieldData ImportData -> Message ImportData -> MessageLength	
Export parameters:	ExportData -> Equipment (equipment number)	
Return parameters:	0 1-9 99	o.k. RFC communication error Application errorr

Creating a Functional Location

You can use this function to create a functional location.

C prototype:	*int CadRfcFloCreate (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_CREATE_FUNC_LOC_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code) CadRfcImportField->FunctionalLocation (functional location)	
Export parameters:	CadRfcExportField->FunctionalLocation (functional location) CadRfcExportField->StructureIndicator (structure indicator) CadRfcExportField->FunctionalLocationType (functional location category) CadRfcExportField->RefReferenceLocation (copy from functional location) CadRfcExportField->RefFunctionalLocation (copy from reference functional location)	
Return parameters:	0 1-9	o.k. RFC communication error

Changing a Functional Location

You can use this function to change a functional location.

C prototype:	*int CadRfcFloChange (CadRfcExportField *, CadRfcImportField *, ITAB_H **,ITAB_H **)*	
RFC module:	RFC_CHANGE_FUNC_LOC_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->FunctionalLocation (functional location) CadRfcExportField->StructureIndicator (structure indicator)	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying a Functional Location

You can use this function to display a functional location.

C prototype:	*int CadRfcFloDisplay (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_FUNC_LOC_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->FunctionalLocation (functional location) CadRfcExportField->StructureIndicator (structure indicator)	
Return parameters:	0 1-9	o.k. RFC communication error

Reading a Functional Location

You can use this function to read functional location data. The fields of the IFLO structure are returned in the *FieldData* table.
You can read an entry by using the *CadRfcGetFieldTabEntry*. The fields in the *FieldData* table are filled as follows:

- *Fieldmulti* (language-dependent short text of the data base field)
- *Fieldname* (field name, for example: MARA-MATNR)
- *Fieldvalue* (value)

C prototype:	*int CadRfcGetFuncLocData (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_GET_FUNC_LOC_DATA	
Import parameters:	ITAB_H FieldData ImportData -> Message ImportData -> MessageLength	
Export parameters:	ExportData -> Functional Location (number of functional location)	
Return parameters:	0 1-9 99	o.k. RFC communication error Application errorr

Creating a Maintenance Notification

You can use this function to create a maintenance notification.

C prototype:	*int CadRfcPmrCreate (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **) ITAB_H **)*	
RFC module:	RFC_CREATE_PM_REQUEST_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code) CadRfcImportField->Notification (notification number)	
Export parameters:	CadRfcExportField->NotificationType (notification type) CadRfcExportField->FunctionalLocation (functional location) CadRfcExportField->Equipment (equipment number) CadRfcExportField->Material (material number)	
Return parameters:	0 1-9	o.k. RFC communication error

Changing a Maintenance Notification

You can use this function to change a maintenance notification.

C prototype:	*int CadRfcPmrChange (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_CHANGE_PM_REQUEST_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->Notification (notification number)	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying a Maintenance Notification

You can use this function to display a functional location.

C prototype:	*int CadRfcPmrDisplay (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_PM_REQUEST_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->Notification (notification number)	
Return parameters:	0 1-9	o.k. RFC communication error

Creating a Maintenance Order

You can use this function to create a maintenance order.

This function is not yet supported in release 3.0.

C prototype:	*int CadRfcPmOrderCreate (CadRfcExportField *,CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_CREATE_PM_ORDER _MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code) CadRfcImportField->PmOrderNo (PM order number)	
Export parameters:	CadRfcExportField->PmOrderType (PM order type) CadRfcExportField->FunctionalLocation (functional location) CadRfcExportField->Equipment (equipment number) CadRfcExportField->Material (material number) CadRfcExportField->Plant (plant) CadRfcExportField->BusinessArea (business area)	
Return parameters:	0 1-9	o.k. RFC communication error

Changing a Maintenance Order

You can use this function to change a maintanance order.

This function is not yet supported in release 3.0.

C prototype:	*int CadRfcPmOrderChange (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_CHANGE_PM_ORDER_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->PmOrderNo (PM order number)	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying a Maintenance Order

You can use this function to display a maintanance order.

This function is not yet supported in release 3.0.

C prototype:	*int CadRfcPmOrderDisplay (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_PM_ORDER_MASTER	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->PmOrderNo (PM order number)	
Return parameters:	0 1-9	o.k. RFC communication error

Mail and Workflow Functions

The following mail and workflow functions are supported:

Displaying a Mail Inbox [Seite 604]

Displaying a Mail Outbox [Seite 605]

Displaying a Workflow Inbox [Seite 606]

Displaying a Mail Inbox

You can use this function to display a mail inbox.

C prototype:	*int CadRfcMailInbox (CadRfcExportField *, CadRfcImportField *,ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_MAIL_INBOX	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	-	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying a Mail Outbox

You can use this function to display a mail outbox.

C prototype:	*int CadRfcMailoutbox (CadRfcExportField *, CadRfcImportField*, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_MAIL_OUTBOX	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	-	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying a Workflow Inbox

You can use this function to display a workflow inbox.

C prototype:	*int CadRfcWorkflowInbox (CadRfcExportField *,CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_WORKFLOW_INBOX	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	-	
Return parameters:	0 1-9	o.k. RFC communication error

General Functions in the SAP Environment

The following general functions are supported:

Displaying a Product Structure [Seite 608]
Requesting SAP System Parameters [Seite 609]
Displaying a Where-Used List for a Material or Document [Seite 610]
Finding Objects via Matchcode [Seite 611]
Finding Objects in Classes [Seite 617]
Return of Last RFC Error Message [Seite 612]
Reading an Entry in Table SAP_FIELD_DATA [Seite 613]
Reading an Entry in Table DMS_CLASS_DATA [Seite 614]
Making an Entry in Table SAP_FIELD_DATA [Seite 615]
Making an Entry in Table DMS_CLASS_DATA [Seite 616]

Displaying a Product Structure

You can use this function to display the structure overview of a material or document by entering the material number or document key, respectively.

C prototype:	*int CadRfcProductStructure (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_EXPLODE_PRODUCT_STRUCTURE	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->DocNumber (document number) CadRfcExportField->Dokar (document type) CadRfcExportField->Doktl (document part) CadRfcExportField->Dokvr (document version) CadRfcExportField->BomApplication (Application) CadRfcExportField->Plant (plant) CadRfcExportField->BomAlternative (alternative BOM) CadRfcExportField->ChangNo (change number) CadRfcExportField->ValidFrom (valid-from date) CadRfcExportField->BomType (BOM usage)	
Return parameters:	0 1-9	o.k. RFC communication error

Requesting SAP System Parameters

You can use this function to request SAP System parameters.

C prototype:	*int CadRfcGetSystemParas (CadRfcImportField *)*	
RFC module:	RFC_GET_SAP_SYSTEM_PARAMETERS	
Import parameters:	CadRfcImportField->DateFormat (date format) CadRfcImportField->DecimalSign (decimal point) CadRfcImportField->Language (logon language) CadRfcImportField->SapSystemRelease (SAP release) CadRfcImportField->UserName (user) CadRfcImportField->Language (logon language)	
Return parameters:	0 1-9	o.k. RFC communication error

Displaying a Where-Used List for a Material or Document

You can use this function to display a where-used list for a material or document by entering the material number or document key, respectively.

C prototype:	*int CadRfcBomWhereUsed (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_DISPLAY_OBJECT_WHERE_USED	
Import parameters:	CadRfcImportField->FunctionCode (function code)	
Export parameters:	CadRfcExportField->Material (material number) CadRfcExportField->DocNumber (document number) CadRfcExportField->Dokar (document type) CadRfcExportField->Doktl (document part) CadRfcExportField->Dokvr (document version) CadRfcExportField->Plant (plant) CadRfcExportField->ValidFrom (valid-from date) CadRfcExportField->ValidTo (valid-to date) CadRfcExportField->SkipFirstScreen (skip initial screen)	
Return parameters:	0 1-9	o.k. RFC communication error

Finding Objects via Matchcode

You can use this function to call the SAP matchcode functionality.

C prototype:	*int CadRfcMatchcode (CadRfcExportField *, CadRfcImportField *,ITAB_H **SapFieldData, ITAB_H **DmsClassData)*	
RFC module:	RFC_SELECT_OBJ_VIA_MATCHCODE	
Import parameters:	CadRfcImportField->FunctionCode (function code) CadRfcImportField->MessageLength (length of the error message) CadRfcImportField->FunctionCode (function code)	
Export parameters:	ITABH** SapFieldData (pointer on table with selected objects) CadRfcExportFields->ReturnMcdValues (return of selected object (“X”) oder go to the pointer (“ “))	
Return parameters:	0 1-9	o.k. RFC communication error

Return of Last RFC Error Message

This function returns the last RFC error message.

C prototype:	*int CadRfcGetLastError (char *ErrorMessage)*	
RFC module:	-	
Import parameters:	char * ErrorMessage	
Export parameters:	-	
Return parameters:	0 - 99	o.k. no error messages

Reading an Entry in Table SAP_FIELD_DATA

CadRfcGetFieldTabEntry returtns the entry with the *EntryNumber* number from the *SapFieldTab* table in the *FieldData* structure. The table is structured according to the SAP_FIELD_DATA structure.

C prototype:	*int CadRfcGetFieldTabEntry (int EntryNumber, ITAB_H *SapFieldTab, SAP_FIELD_DATA *FieldData)*	
RFC module:	-	
Import parameters:	SAP_FIELD_DATA	
Export parameters:	int EntryNumber (number of the entry in the SapFieldTab table) ITAB_H * SapFieldTab (points to internal table)	
Return parameters:	0 1-9 99	o.k. RFC communication error Application errorr

Reading an Entry in Table DMS_CLASS_DATA

CadRfcGetCharTabEntry returns the entry with the *EntryNumber* number from the *SapCharacTab* table in the *CharacData* structure. The table is structured according to the CLASS_CHARACTERISTICS structure.

int CadRfcGetDmsClassTabEntry ;

CadRfcGetFieldTabEntry returtns the entry with the *EntryNumber* number from the *SapFieldTab* table in the *FieldData* structure. The table is structured according to the SAP_FIELD_DATA structure.

C prototype:	*int CadRfcGetDmsClassTabEntry (int EntryNumber, ITAB_H *thSapCharacTab, CLASS_CHARACTERISTICS* CharacData)*	
RFC module:	-	
Import parameters:	CLASS_CHARACTERISTICS (a table entry)	
Export parameters:	int EntryNumber (line of table to be read) ITAB_H * theSapCharacTab(points to table)	
Return parameters:	0 227-230	o.k. Table processing errors

Making an Entry in Table SAP_FIELD_DATA

A line of the table SAP_FIELD_DATA is filled out by repeating the function call for this function. Each time the table is called, you can transfer an attribute value for an attribute name. The index transferred defines the table line. This function is currently only used to transfer object links to the SAP System with functions *CadRfcDocCreate* and *CadRfcDocChange*

The first time you call this function with index 1, the entire table is initialized.

The attribute names match the names of the SAP_FIELD_DATA structure defined in header file *cadrfc.h*.

The index must be counted up sequentially. If the index changes, the system assumes that the current line is filled out and automatically inserts the line in the table. You can then no longer change the line. If you try to access the line later using a lower index, you see an error message.

To close the function for filling out the BOM item table, use a function call with index “0”.

C prototype:	*int CadRfcWriteFieldTabEntry* (char*AttributeName, char*AttributeValue, int EntryNumber);	
RFC module:	-	
Import parameters:	-	
Export parameters:	char *AttributeName char *AttributeValue int EntryNumber	
Return parameters:	0 232-234	o.k. Table processing errors

Making an Entry Table DMS_CLASS_DATA

A line of the table DMS_CLASS_DATA is filled out by repeating the function call for this function. Each time the table is called, you can transfer an attribute value for an attribute name. The index transferred defines the table line.

The first time you call this function with index 1, the entire table is initialized.

The attribute names match the names of the CLASS_CHARACTERISTICS structure defined in header file *cadrfc.h*.

The index must be counted up sequentially. If the index changes, the system assumes that the current line is filled out and automatically inserts the line in the table. You can then no longer change the line. If you try to access the line later using a lower index, you see an error message.

To close the function for filling out the BOM item table, use a function call with index “0”.

C prototype:	*int CadRfcWriteCharTabEntry (char *AttributeName, char *AttributeValue, int EntryNumber);*	
RFC module:	-	
Import parameters:	-	
Export parameters:	char *AttributeName char *AttributeValue int EntryNumber	
Return parameters:	0 232-234	o.k. Table processing errors

Finding Objects in Classes

You can use this function to call the SAP function for finding objects via classes. This function is supported for all classifiable objects.

C prototype:	*int CadRfcClassSelection (CadRfcExportField *, CadRfcImportField *, ITAB_H **, ITAB_H **, ITAB_H **)*	
RFC module:	RFC_SELECT_OBJECTS_VIA_CLASS	
Import parameters:	CadRfcImportField->FunctionCode (function code) CadRfcImportField->ClassNo (class number) CadRfcImportField->ClassType (class type) CadRfcImportField->ObjectTable (name of database table) CadRfcImportField->ObjectKey (object key) CadRfcImportField->ValidFrom (valid-from date) ITAB_H ** (pointer to table with characteristics data)	
Export parameters:	CadRfcExportField->ClassNo (class number) CadRfcExportField->ClassType (class type) CadRfcExportField->ValidFrom (valid-from date)	
Return parameters:	0 1-9	o.k. RFC communication error

If a document is selected for insertion or checked out during this function, a table with characteristics data is made available by the RFC module. A pointer to this table is returned via the second ITAB_H parameter. The structure of the table is defined as *CLASS_CHARACTERISTICS* in the header file *cadrfc.h.*

Structure of *CLASS_CHARACTERISTICS.*

Atnam	Name of the characteristic
Atbez	Description of characteristic
Atwrt	Characteristic value
Dinkb	DIN code letter
Unit	Unit of measure

The object is returned as follows:

- If **one** object is selected in the object list and you execute the *Copy f*unction, the selected object is returned with the CadRfcImportField->ObjectKey import parameter. You can find out the object type from the ObjectTable parameter. Additionally, the object is

returned in the SapFieldData table. The characteristic data are returned in the second ITAB_H parameter.

- If **more than one** object is selected, the table is not filled with the characteristic data. Only a list of the objects in the SapFieldData table is returned. The structure of this table is defined as SAP_FIELD_DATA in the cadrfc.h header file.

  The table has the following structure:

  – *FieldMulti* (currently not filled)

  – *FieldName* (data base table)

  – *FieldValue* (object key)

See also:

Example of a Call [Seite 619]

Example of a Call

```
#include "cadrfc.h"
ITAB_H            *FldDataPtr;
ITAB_H            *DmsCharacPtr;
ITAB_H            *ClsCharacPtr;
RFC_RC            CadRfcRc;
CadRfcExportField *RfcExport;
CadRfcImportField *RfcImport;
CLASS_CHARACTERISTICS *ClsCharac;
SAP_FIELD_DATA SapField     Data;
int  I=0;
char lAttributName (31);    char lFieldName (36);
void *pLine;
int index = 0;
CadRfcRc =  CadRfcClassSelection(RfcExport, RfcImport,
            &FldDataPtr, &ClsCharacPtr, &DmsCharacPtr);
/* Tabelle mit Merkmalsdaten Zeile fuer Zeile einlesen */
index = 1;
do {
   I = 0 CadRfcGetCharTabEntry (index, ClsCharacPtr, &ClsCharac);
    if (I = 0) {
        memcpy (lAttributName, ClsCharac, Atname,30);
      ...
      ...
    }
    index ++;
  }
```

Using Calls in the SAP-CAD Function Library

When working with the dialog RFC interface, you can also use all API calls of the CAD dialog interface at any time.

You do not need to establish a second connection to the R/3 System. The connection established between the R/3 System and the external application serves both purposes.

If you want to use function calls of the SAP-CAD function library, please note the following points:

- You must install the library *caddialg.o* or *caddialg.lib*.

  To enable you to use the API calls of the CAD dialog interface, both the RFC library and the SAP-CAD function library *caddialg.o* or *caddialg.lib* must be installed in your own interface program.

- You must initialize the SAP-CAD function library.

  Before you can make the first API call from the SAP-CAD function library, an initialization must take place. This is done with the following function call:

  hd = SapConncRfc(argv, cadsys, rcvstring, msg)

  Key:

SAPHD hd	Structure of type SAPHD Description of Function Library caddialg [Seite 310]
char *argv[]	Array pointer containing the program path and program name in argv[0] (Example: /usr/sap/cadtst1)
char *cadsys	External system
char **rcvstring	Data string with status information
char **msg	Message received from SAP System

  The structure of the information returned from the SAP System via *rcvstring* matches the structure of the information in the system function SYSRQ PA (*Display system parameters*).

Using the CAD Desktop

Use

The CAD Desktop is a tool that simplifies the integration of CAD systems for processing document structures.

- These document structures are created in the CAD system when assemblies and part files are created (Original application files for assemblies and the parts belonging to them) and are usually modulated in the SAP System using document structures.
  The individual parts of the document structure (BOM headers and items) represent the assembly and part files contain addition administrative data (such as person responsible, status).
- After the CAD data is transferred into the SAP System you can access this data when processing the document structure. The constructor or engineer also has an easy-to-use environment for carrying out various processing activities such as:
  - Load all original application files that are dependent on a change into the CAD system in **one** step The engineer processes the original application files in the CAD system.
  - After making the changes the engineer can transfer all the original application files back into the SAP System in **one** step. In this phase changes are possible for documents such as status changes.

Integration

The CAD Desktop allows you to control the processes. You can decide, for example, whether and in which environment the implemented routines from the standard SAP System are used and whether special enterprise-specific processes are implemented with other programs.

You can use all functions and features of the standard SAP System by integrating the CAD Desktop:

- Generate the programming interface of the CAD Desktop from the SAP System.

  Call in the standard SAP System: *Tools* → *ABAP Workbench Development* → *Function Builder* (development class CA, function group CAD_DESKTOP, Function module CAD_DESKTOP).
- Integrate the coding into your CAD application. Use the RFC program library that can be found under the following path when a full installation of the SAP System was made: *Start* → *Programs* → SAP *Automation* → *SAP Automation Help*.

You can integrate enterprise specific processes as follows:

- Add to the context menu by means of individual configurations of the CAD Desktop.
- Use the ***SAP Enhancement concept*** functions.
  The Exits provided have no function. They only represent starting points for planned functions of the CAD interface. These functions can be set up with enterprise-specific logic.

  See also: Integrating Enterprise Specific Processes [Seite 624]

Features

Call

The initial screen can be obtained via a file structure. In this way you have the possibility to manage documents that are not normally managed using the SAP System.

When document structures in the SAP System are used for management you can enter the identifying key data of the document (document of the BOM header). If you do not know any document data a screen appears where you can find the document of the BOM header as follows:

- Enter the file structure when you want to process a document structure that is not managed using the SAP System.
- If you want to process a document structure that is stored in the SAP System enter the document key from the header of the document structure. Additional entries for effectivity are supported.
- If you do not enter any data of the document structure you want to process the system starts the transaction CV04N, matchcode search automatically for finding documents in the SAP System.

When you call the CAD Desktop without document information you will see a screen where you can find document of the BOM header as follows:

- Stack with the last ten document that you processed
- Document search of the SAP Systems (transaction CV04N)
- Matchcode search

After you have entered the document of the BOM header the document structure is exploded single-level. The data is returned that was valid for a particular effectivity date, date of a change number, or a revision level.

Functions

The CAD Desktop is a tool for processing document structures. The following overview shows all available functions.

Function	Use
Mass operations	You can select one or more documents and execute one for the following functions: • Check in documents with or without status change • Check out documents for viewing and processing • Change the document status and person responsible

Functions for processing the structure	• Display identical documents in the structure • Management of display variants • Print the structure • Search within the structure Within the structure you can display the dependencies of a document to other objects by expanding the entries; example *Document* → *Originals* → *User data* → *Versions* → *Materials* By double clicking on the document key you can display the document (transaction CV03N). If the material link is not clear or incorrect a symbol of a stack appears. Click on the stack to produce a selection list.
General Functions	• Calling the Product Structure Browser • Refresh • Single-level explosion of the document structure

Activities

You can access the initial screen from the document structure by using known data as follows:

- Enter the file structure when you want to process a document structure that is not managed using the SAP System.
- If you want to process a document structure that is stored in the SAP System enter the document key from the header of the document structure. Additional entries for effectivity are supported.
- If you do not enter any data of the document structure you want to process the system starts the transaction CV04N, matchcode search automatically for finding documents in the SAP System.

Integrating Enterprise-Specific Processes

Use

You use Customer-Exits [Extern] to integrate enterprise-specific processes. The following Exists are supported by CAD Desktop:

- Menu-Exits

  Each menu-exit corresponds to a function-exit.

- Function-exits
- Screen-Exit

  The screen-exit corresponds to three function-exits.
  In order to insert a sub-screen area automatically in to the main screen section of the CAD-Desktop you must have data transfer possible in both directions. For this you need two function-exits. The third is needed in order to process the OK-Code of the sub-screen-exits in PAI of the main screen.

The function module is in the function group XCADDESKTOP (development class CA). You can use the enhancement of the SAP Enhancement concept and then activate it.

The following overview is a list of the enhancements as well as the supported exits.

Enhancement project	Supported exits
CADD0001	Function-Exit for function button 1 and menu-exit
CADD0002	Function-Exit for function button 3 and menu-exit
CADD0003	Function-Exit for function button 2 and menu-exit
CADD0004	Function-exits for • Data import • Processing the function code (screen area) • Data export Screen-Exit

Procedure

Create an enhancement project by entering the enhancement that was provided by SAP. The enhancement contains the function module for the desired function. The function modules branch to an include program that you encode in the enterprise-specific function requests.

See also: Creating Additional Projects [Extern]

Appendix

This section contains information on the following topics:

Error messages in function library caddialg [Seite 626]

CPIC Communication Errors and CPIC Error Codes [Seite 628]

Return Codes from the SAP System [Seite 630]

Return codes of the SAP-CAD function library [Seite 631]

Return codes in the cadrfc00 library [Seite 632]

File sideinfo [Seite 635]

File saprfc.ini [Seite 636]

File cadrfc.ini [Seite 643]

File caddialg.h [Seite 653]

File cadrfc.h [Seite 656]

File cadtst1.c [Seite 670]

Error messages in function library *caddialg*

Function library *caddialg* contains the following error messages:

Error messages in function library *caddialg*

Error message	Description
EM001	**Missing data-record in file cademu.dat!** No response string was found in file cademu.r3 for the string transferred. Possible cause of error: missing dividing line to define beginning/end of response string.
EM003	No entry found in file cademu.dat No suitable entry was found in file cademu.r3 for the string transferred. Using log file log_xxxx.log if necessary, check whether the string transferred matches the entry in cademu.r3 byte for byte. The first 60 bytes of the string are compared.
EM004	**Emulationfile cademu.r3 not found!** File cademu.r3 was not found. This file must be in the directory in which the interface program was started.
EM005	**More data-records available for last function!** A new function was called before all the data of the last function had been retrieved.
FB002	Unable to write to file sap_cad.log File sap_cad.log could not be accessed (authorization problem?)
FB003	Memory allocation failure for hd.cpicmes The program could not allocate memory to variable hd.cpicmes.
FB004	Unable to create logfile for TRACESTD option Log file log_xxxx.log cannot be created (authorization problem?)
FB005	**File sapmm29c.ini not found.** Configuration file caddialg.ini was not found. This file must be in the directory in which the interface program was started.
FB006	CPIC-Error by XXXXXX CPIC communication error. 'XXXXXX' stands for the CPIC function in which the error occurred. (see: CPIC Communication Errors and CPIC Error Codes [Seite 628])
FB007	Memory allocation failure for buffer The program could not allocate memory to variable buffer. One cause of this error could be that you want to transfer very large quantities of data between the external application and the R/3 System.
FB102	Memory allocation failure for sbuf The program could not allocate memory to variable sbuf. One cause of this error could be that you want to transfer very large quantities of data between the external application and the R/3 System.
FB103	Unable to write to file You want to check out an original application file from the SAP System. The file cannot be created on the local computer.
FB104	Unable to read from file You want to check in an original application file to the SAP System. The file cannot be accessed.

FB106	No file name You want to check in an original application file to the SAP System. However, you have not entered a file name.
FB116	64K data allowed with MS-Windows In MS Windows, no more than 64KB of data can be transferred in one function.
FB122	SAPFTP could not be started on your PC Program SAPFTP.EXE or SAPFTP (UNIX) could not be started on the local computer.
FB123	Program SAPFTP not found on your PC Program SAPFTP.EXE or SAPFTP (UNIX) does not exist on the local computer. For information on the directories in which this program could be stored, see: Check In for Original Application Files [Seite 166]
FB124	**SAPFTP could not be registered** Program SAPFTP could not be registered on the application server.
FB125	**Host name for the local computer not found** The local host name could not be determined.
FB126	Maximum length of file name is 70 bytes The name of a file to be checked in or out is too long. The name cannot contain more than 70 characters.
FB127	File could not be created An original file is to be checked out of the SAP System. However, the file could not be created locally.
FB1268	**SAPFTP could not be registered. Maximum length of file name is 70 bytes** The DVA (Direct Vault Access) registration was incorrect. A trace file *dev_ftp* can be created if you set the environment variable FTP_TRACE 0 2. A log file is kept in this file of the registration errors that have occurred. The name of a file (to be checked out) is too long. The name cannot contain more than 70 characters.
FB129	**Error working in emulation-mode.** **Check if CadDialogEmulationMode in file CADRFC.INI is set to 1.** You are working in emulation mode. You must set the *CadDialogEmulationMode* parameter in the CADRFC.INI file to 1.

CPIC Communication Errors and CPIC Error Codes

CPIC communication errors 'XXXXXX'

The following CPIC communication errors can occur:

CPIC communication errors

Error	Description
CMINIT	Initialize connection
CMALLC	Establish connection
CMRCV	Receive data
CMSEND	Send data
CMDEAL	Terminate connection
CMCNVO	Convert from ASCII to EBCDIC
CMCNVI	Convert from EBCDIC to ASCII

CPIC error codes

The following CPIC error codes can occur:

CPIC error codes

Code	Description
1	CM_ALLOCATE_FAILURE_NO_RETRY
2	CM_ALLOCATE_FAILURE_RETRY
3	CM_CONVERSATION_TYPE_MISMATCH
6	CM_SECURITY_NOT_VALID
8	CM_SYNC_LVL_NOT_SUPPORTED_PGM
9	CM_TPN_NOT_RECOGNIZED
10	CM_TP_NOT_AVAILABLE_NO_RETRY
11	CM_TP_NOT_AVAILABLE_RETRY
17	CM_DEALLOCATED_ABEND
18	CM_DEALLOCATED_NORMAL
19	CM_PARAMETER_ERROR
20	CM_PRODUCT_SPECIFIC_ERROR
21	CM_PROGRAM_ERROR_NO_TRUNC
22	CM_PROGRAM_ERROR_PURGING
23	CM_PROGRAM_ERROR_TRUNC
24	CM_PROGRAM_PARAMETER_CHECK

25	CM_PROGRAM_STATE_CHECK
26	CM_RESOURCE_FAILURE_NO_RETRY
27	CM_RESOURCE_FAILURE_RETRY
28	CM_UNSUCCESSFUL
35	CM_OPERATION_INCOMPLETE
36	CM_SYSTEM_EVENT

For more detailed information on these error codes, please refer to the relevant CPIC literature.

Return Codes from the SAP System

Each time you call a function from the SAP-CAD function library, a return code is entered in structure *SAPHD,* field *subrc*. This return code shows you whether an error has occurred during processing in the SAP System.

If the content of this field is not zero, an error has occurred, and the function concerned is terminated by the SAP System. If this is the case, no data is returned. In the message field of the function call, you see information on the type of error.

Return codes of the SAP-CAD function library

Each time you call a function from the SAP-CAD function library, a return code is entered in structure *SAPHD,* field *cpicrc*. This return code shows you whether an error that makes further processing impossible has occurred, either during communication with the R/3 System or within the SAP-CAD function library. In both cases, the connection to the R/3 System is terminated.

The following return codes are supported:

Return codes of the SAP-CAD function library

Returncode	Description
1-36	CPIC communication error (see FB006 in Error Messages in Function Library caddialg [Seite 626])
100	CPIC communication error (see FB007 in Error Messages in Function Library caddialg [Seite 626])
101	CPIC communication error (see FB003 in Error Messages in Function Library caddialg [Seite 626])
102	CPIC communication error (see FB102 in Error Messages in Function Library caddialg [Seite 626])
103	CPIC communication error (see FB103 in Error Messages in Function Library caddialg [Seite 626])
104	CPIC communication error (see FB104 in Error Messages in Function Library caddialg [Seite 626])
105	CPIC communication error (see FB105 in Error Messages in Function Library caddialg [Seite 626])
110	CPIC communication error (see FB110 in Error messages in function library caddialg [Seite 626])
111	No memory can be allocated to local variable tembuf1 in routine CadRfcReceive.
112	RFC error in RFC call RfcCallReceive in routine CadRfcSend
113	RFC error in RFC call RfcCallReceive in routine CadRfcReceive
114	No memory can be allocated to local variable tembuf in routine CadRfcReceive.
115	No memory can be allocated to local variable tembuf in routine CadRfcSend.
116	CPIC communication error (see FB116 in Error messages in function library caddialg [Seite 626])
118	Error occurred when allocating memory to thBinTable. No memory could be allocated to structure thBinTable (internal structure for a binary table) from the program.
119	Line could not be added to table thBinTable. A line was to be added to table thBinTable with an RFC call. An error has occurred.
120	CPIC communication error (see FB002 in Error messages in function library caddialg [Seite 626])
500	No memory could be allocated to temporary data when working with the emulation library.

Return Codes in the cadrfc00 Library

The table below lists all the return codes in the cadrfc00 library.

Return codes in the cadrfc00 library

Returncode	Description
1 2 3 4 5 6 7 8 9	**RFC_FAILURE Error occurred** **RFC raised** **RFC_SYS_EXCEEPTION System exception raised, connection closed** **RFC_CALL Call received** **RFC_INTERNAL_COM Internal communication, repeat** **RFC_CLOSED Connection closed by the other side** **RFC_RETRY No Data yet (RfcListen only)** **RFC_NO_TID No Transaction ID available** **RFC_EXECUTED Function already executed**
200	**File *cadrfc.ini* not found** *CadRfcInit* was called, but no arguments were transferred. The program now assumes that the reconnect parameters are stored in file *cadrfcid.tmp*. The program tries to determine the storage location of the file from file *cadrfc.ini*. However, file *cadrfc.ini* could not be found. The program looks for file *cadrfc.ini* in the path stored in environment variable CADRFC first.
201	**Entry *CadRfcTempDir* not found in file *cadrfc.ini*** *CadRfcInit* was called, but no arguments were transferred. The program now assumes that the reconnect parameters are stored in file *cadrfcid.tmp*. The program tries to determine the storage location of the file from file *cadrfc.ini*. However, entry *CadRfcTempDir* is missing in the file.
202	**File *cadrfcid.tmp* not found** *CadRfcInit* was called, but no arguments were transferred. The program now assumes that the reconnect parameters are stored in file *cadrfcid.tmp*. However, file *cadrfcid.tmp* was not found in the path entered in file *cadrfc.ini* under parameter *CadRfcTempDir*.
203	***RfcAccept* does not return a handle** In the *CadRfcInit* call, an *RfcAccept* call was executed from the standard SAP-RFC library. No communication handle could be returned for the parameters transferred.
204	**No RFC connection currently exists** You tried to execute an RFC function, but no RFC connection currently exists.
205	**Error allocating memory to MaterialHeaderData** The program could not allocate memory to structure *MaterialHeaderData*.
206	**Error allocating memory to pExportData** The program could not allocate memory to structure *pExportData*.
207	**Error allocating memory to pImportData** The program could not allocate memory to structure *pImportData*.
208	**Error allocating memory to thSapFieldData** The program could not allocate memory to structure *thSapFieldData*.

209	**Error allocating memory to thClsCharac** The program could not allocate memory to structure *thClsCharac*.
210	**Error allocating memory to thDmsCharac** The program could not allocate memory to structure *thDmsCharac*.
211	**Error allocating memory to pDocumentData** The program could not allocate memory to structure *pDocumentData*.
212	**Line could not be inserted in table thItemData** You tried to add a line to table *thItemData* in a standard RFC function call. An error has occurred.
213	**Table thItemData could not be created** You tried to create table *thItemData* in a standard RFC function call. An error has occurred.
214	**Error allocating memory to ItemData** The program could not allocate memory to structure *ItemData*.
215	**File *cadrfc.ini* not found** The configuration file was searched in the following sequence: 1. Path under which environment variable CADRFC is stored 2. Current directory
216	**Entry not found in file *cadrfc.ini*** A required parameter for API call *CadRfcOpen* was not found in configuration file *cadrfc.ini*. Please check whether this file is complete.
217	**Error calling *RfcOpen*** No connection could be established to R/3. For more information on why the function was terminated, see RFC trace file *dev_rfc*.
218	**CadRfcGetBomItemEntry: Error on parameter transferral** The storage space for structure CSRFCITEM has not been correctly allocated.
219	**CadRfcGetBomItemEntry: Internal table has no entry** There is no entry in the transferred table.
220	**CadRfcGetBomItemEntry: Line number is invalid** The selected line does not exist in the internal table.
221	**CadRfcGetBomItemEntry: Error on reading internal table** An error has occurred on reading a table entry.
223	**Configuration parameters have not been maintained in CADRFC.INI** The configuration parameters CadRfcHostname und CadRfcSysnr have not been set.
224	**Line could not be added to table SAP_FIELD_DATA** An error has occurred on adding a new line to table SUB_ITEM_DATA.
225	**Error on allocating memory to SUB_ITEM_DATA** No memory could be allocated to structure SUB_ITEM_DATA from the program.
226	**Error on allocating memory to SubItemData** No memory could be allocated to the structure SubItemData from the program.
227	**CadRfcGetCharEntry: Error on parameter transferral** The storage space for structure CSSUBITEM has been incorrectly allocated.
228	**CadRfcGetCharTabEntry: Internal table has no entry** There is no entry in the transferred table.
229	**CadRfcGetCharTabEntry: Line number is invalid** The selected line does not exist in the internal table.
230	**CadRfcGetCharTabEntry: Error on reading internal table** An error has occurred on reading a table entry.

Return Codes in the cadrfc00 Library

232	**Line could not be added to table thClsCharac** A line is to be added to table thClsCharac with an RFC call. The selected line does not exist in the internal table. An error has occurred.
233	**Error on allocating memory to pSapFieldData** No memory could be allocated to structure pSapFieldData from the program.
234	**Error on allocating memory to pClsCharac** No memory could be allocated to structure pClsCharac from the program.
235	**Line could not be added to table thSapFieldData** An error has occurred on adding a line to table thSapFieldData.

File *sideinfo*

DEST=K50

LU=K50

TP=*1SA

GWHOST=is0001

GWSERV=sapgw00

PROTOCOL=C

CPIC_TRACE=3

File *saprfc.ini*

```
/*********************************************************************/
/*                                                                   */
/*                        SAPRFC.INI                                 */
/*                                                                   */
/*                                                                   */
/*  The 'saprfc.ini' file must be in the same directory as the RFC   */
/*  client or server program or it can completely be defined by the  */
/*  environment variable RFC_INI with a full path and file name.     */
/*  Example on WINDOWS: set RFC_INI=d:\rfctest\saprfc.ini.           */
/*  Example on AS/400: ADDENVVAR ENVVAR(RFC_INI) VALUE('/QSYS.LIB/   */
/*                     R340ARFC.LIB/INI.FILE/SAPRFC.MBR')            */
/*                                                                   */
/*  See description at the end of this file.                         */
/*                                                                   */
/*********************************************************************/


/*===================================================================*/
/*  Type B:  R/3 system - load balancing feature                     */
/*===================================================================*/
DEST=BIN
TYPE=B
R3NAME=BIN
MSHOST=hs0311
GROUP=PUBLIC
RFC_TRACE=0
ABAP_DEBUG=0
USE_SAPGUI=0

DEST=BIND
TYPE=B
R3NAME=BIN
```

```
MSHOST=hs0311
GROUP=PUBLIC
RFC_TRACE=0
ABAP_DEBUG=1
USE_SAPGUI=0

DEST=BING
TYPE=B
R3NAME=BIN
MSHOST=hs0311
GROUP=PUBLIC
RFC_TRACE=0
ABAP_DEBUG=0
USE_SAPGUI=1

DEST=BINS
TYPE=B
R3NAME=BIN
MSHOST=hs0311
GROUP=PUBLIC
RFC_TRACE=0
ABAP_DEBUG=0
USE_SAPGUI=0
SNC_MODE=1
SNC_PARTNERNAME=p:CN=bin.hs0311, OU=TEST, O=SAP, C=DE
SNC_LIB=/sapmnt/is0003/l/secude/hp/lib/libsecude.sl
*SNC_LIB=\\hw1329\snc

/*=====================================================================*/
/*  Type A:  R/3 system - specific application server              */
/*=====================================================================*/
DEST=BIN_HS0011
TYPE=A
ASHOST=hs0011
SYSNR=53
```

File saprfc.ini

```
RFC_TRACE=0
ABAP_DEBUG=0
USE_SAPGUI=0

/*=====================================================================*/
/*  Type A:  R/3 system - specific application server              */
/*  Working with SNC                                               */
/*=====================================================================*/
DEST=BIN_HS0019
TYPE=A
ASHOST=hs0019
SYSNR=53
RFC_TRACE=0
SNC_MODE=1
SNC_PARTNERNAME=s:sap53@hs0019
SNC_LIB=/krb5/hpux/lib/libkrb5.sl
*SNC_LIB=e:/krb5/krb5-32.dll

/*********************************************************************/
/*                                                                   */
/*                        SAPRFC.INI                                 */
/*                                                                   */
/*                                                                   */
/*  Depending on the programming in RFC client or server program     */
/*  the RFC library on an external system such as Windows or         */
/*  any R/3-based system can read this file to find out all          */
/*  connection type and RFC specific parameters needed to connect to */
/*  a R/2 or R/3 or to register a RFC server program at a SAP gateway*/
/*  and wait for RFC call from a R/2 or R/3 system.                  */
/*                                                                   */
/*  The advantage of this file is all RFC specific parameters known  */
/*  at this time (load balancing, ABAP-Debug, RFC with SAPGUI) and   */
/*  in the future can be used without changes in RFC programs.       */
/*                                                                   */
/*  RFC client programs:                                             */
```

```
/*  ====================                                    */
/*  For using this file RFC client programs must issue the RfcOpenEx */
/*  with DEST=<destination> or the RfcOpen with RFC_MODE_PARAMETER   */
/*  as 'mode'-parameter and the destination must point to an entry   */
/*  of type B, A, 2, E or R in this file.                    */
/*  A sideinfo is no longer needed using this file.                */
/*                                                  */
/*  RFC server programs:                                    */
/*  ====================                                    */
/*  For using this file RFC server programs must issue the RfcAccept */
/*  with -D<destination> as parameter and the destination must point */
/*  to an entry of type R in this file.                      */
/*                                                  */
/*                                                  */
/*  ATTENTION:                                          */
/*  ==========                                          */
/*                                                  */
/*  1. The parameters ABAP_DEBUG and USE_SAPGUI are only possible if */
/*     the RFC server is an R/3 from 3.0C onwards and it will not    */
/*     work with the 16-bit RFC library on Windows nor on AS/400.    */
/*     Moreover, the feature "INVISIBLE SAPGUI" after each           */
/*     RFC-function (USE_SAPGUI=2) requires R/3 from 3.0F onwards    */
/*     and only available with SAPGUI on Windows from 3.0E onwards   */
/*     or SAPGUI on UNIX from 3.1G onwards.                          */
/*                                                  */
/*                                                  */
/*  2. LOAD BALANCING is only available for R/3 from 3.0 onwards.    */
/*                                                  */
/*                                                  */
/*  3. Using sapmsg.ini/saproute.ini of SAPLOGON                    */
/*     If working with Load Balancing and there is no info about     */
/*     MSHOST, the RFC library will try to get this host name from   */
/*     the sapmsg.ini customized for SAPLOGON on Windows.            */
/*     Usually, these files are installed in the Windows directory.  */
/*     You can also copy these files in a directory which is         */
```

File saprfc.ini

```
/*    specified by the environment 'RFC_LOGON_INI_PATH'            */
/*    On NON-Windows platforms, you can work with this environment  */
/*    or copy these files in your working directory.               */
/*                                                                 */
/*    The SAProuter string if exists, must be defined in the R/3    */
/*    system name.                                                 */
/*                                                                 */
/*    e.g.  R3NAME=/H/sapgate1/S/3297/LOI where LOI is the R/3 name.*/
/*                                                                 */
/*    or    R3NAME="/SAP America/LOI" where 'SAP America' is the    */
/*          router name defined in 'saproute.ini' for SAPLOGON on   */
/*          Windows.                                               */
/*                                                                 */
/*                                                                 */
/*  4. connection types:                                           */
/*                                                                 */
/*    Type B: Connect to a R/3 system and the application server    */
/*            will be determined at run time (using the LOAD        */
/*            BALANCING feature of R/3 3.0).                       */
/*            Following parameters can be used:                     */
/*                                                                 */
/*              DEST=<destination in RfcOpen>                       */
/*              TYPE=<B: use load balancing feature>                */
/*              R3NAME=<Name of R/3 system, opt., def.: destination>*/
/*              MSHOST=<Host name of the message server>            */
/*              GROUP=<Appl. servers group name, opt., def.: PUBLIC>*/
/*              RFC_TRACE=<0/1: OFF/ON, optional, default: 0 (OFF)> */
/*              ABAP_DEBUG=<0/1: OFF/ON, optional, default: 0 (OFF)>*/
/*              USE_SAPGUI=<0/1/2: OFF/ON/INVISIBLE_SAPGUI after    */
/*                          each RFC-Function, def.: 0 (OFF)>*/
/*              SNC_MODE=<0/1: OFF/ON, optional, default: 0 (OFF)>  */
/*              SNC_QOP=<1/2/3/8/9, optional, default: 8>           */
/*              SNC_MYNAME=<Own SNC name, optional>                 */
/*              SNC_PARTNERNAME=<SNC name of the message server>    */
/*              SNC_LIB=<Path and file name of the SNC library>     */
```

```
/*                                                    */
/*          The host name and service name of the message server  */
/*          must be defined in the 'hosts' and 'service' files.   */
/*          (<service name> = sapms<R/3 system name>)             */
/*                                                    */
/*          If SNC_MODE is ON the SNC_LIB must be defined in this */
/*          entry or via the environment SNC_LIB.                 */
/*                                                    */
/*                                                    */
/*    Type A: Connect to a specific R/3 application server        */
/*          Following parameters can be used:                     */
/*                                                    */
/*            DEST=<destination in RfcOpen>                       */
/*            TYPE=<A: RFC server is a specific R/3 appl. server> */
/*            ASHOST=<Host name of a specific R/3 appl. server>   */
/*            SYSNR=<R/3 system number>                           */
/*            GWHOST=<optional, default: gateway on appl. server> */
/*            GWSERV=<optional, default: gateway on appl. server> */
/*            RFC_TRACE=<0/1:  OFF/ON, optional, default: 0 (OFF)>*/
/*            ABAP_DEBUG=<0/1: OFF/ON, optional, default: 0 (OFF)>*/
/*            USE_SAPGUI=<0/1/2: OFF/ON/INVISIBLE_SAPGUI after    */
/*                        each RFC-Function, def.: 0 (OFF)>*/
/*            SNC_MODE=<0/1: OFF/ON, optional, default: 0 (OFF)>  */
/*            SNC_QOP=<1/2/3/8/9, optional, default: 8>           */
/*            SNC_MYNAME=<Own SNC name, optional>                 */
/*            SNC_PARTNERNAME=<Partner SNC name>                  */
/*            SNC_LIB=<Path and file name of the SNC library>     */
/*                                                    */
/*          The host name and service name of the specific        */
/*          application server must be defined in the 'hosts' and */
/*          'service' files                                       */
/*          (<service name> = sapdp<R/3 system number>)           */
/*                                                    */
/*          The host name and service name of the SAP gateway must*/
/*          be defined in the 'hosts' and 'service' files. If you */
```

File saprfc.ini

```
/*          don't define GWHOST and GWSERV in this entry, the     */
/*          service name of the SAP gateway still need to be      */
/*          defined in the 'service' file.                        */
/*          (<service name> = sapgw<R/3 system number>)           */
/*                                                                */
/*          If SNC_MODE is ON the SNC_LIB must be defined in this */
/*          entry or via the environment SNC_LIB.                 */
/*                                                                */
/*                                                                */
/*                                                                */
/*  5. Which connection type is to be used ?                      */
/*                                                                */
/*    Type B is recommended for R/3 system (load balancing)       */
/*    Type A is only to be used if you want to connect to a       */
/*            specific application server.                        */
/*                                                                */
/*****************************************************************/
```

File *cadrfc.ini*

```
************************************************************************
**                                                                    **
**                    SAP - CAD Interface 4.0                         **
**                                                                    **
**                      Configuration File                            **
**                                                                    **
**--------------------------------------------------------------------**
**                                                                    **
** The 'cadrfc.ini' file must be in the same directory as the CAD     **
** interface program or the path of this files can be set with the    **
** environment variable CADRFC.                                       **
** Example on WINDOWS: set CADRFC=c:\cad\sample                       **
**                                                                    **
** This configuration file is used when one of the functions          **
** CadRfcOpen or SapConnc is called.                                  **
**                                                                    **
** The configuration file CADDIALG.INI (used in previous versions     **
** of the CAD interface) is not used any more !                       **
**                                                                    **
** At the end of this file you can find detailed information about    **
** the configuration parameters.                                      **
**                                                                    **
************************************************************************

*-----------------------------------------------------
* SAP logon parameters
*-----------------------------------------------------

CadRfcUser                              saptest
CadRfcPassword                          demo
CadRfcClient                            000
CadRfcLanguage                          D
```

File cadrfc.ini

```
*-----------------------------------------------------
* Connection parameters
*-----------------------------------------------------

* Logical destination
CadRfcDestination                        ALR

* SAP-Hostname and System Id
CadRfcHostname
CadRfcSysnr

* Message server name
CadRfcUseSapMsgFile                      Y
CadRfcMessageServer
CadRfcGroup                              PUBLIC

*-----------------------------------------------------
* Parameters specific for Dialog interface (SapConnc)
*-----------------------------------------------------

* SAP version R/3 or R/2
CadDialogSapVersion                      R/3

* CAD system name
CadDialogCadSystem                       DEMO

* net address of the PC
* Needed for document management function. The hostname of the local PC
* could be set with this parameter. If it is not set the environment
* variable HOSTNAME or DISPLAY is read.
CadDialogNetAddress

* Emulation mode
* To work with the emulation mode set this parameter to 1 and
* copy cadcpemu.dll to caddialg.dll (Windows, Windows NT/95)
```

```
* or link your application with cadcpemu.o (UNIX)
CadDialogEmulationMode                  0

*-----------------------------------------------------
* TRACE parameters       0 - set trace off
*                        1 - set trace on
*-----------------------------------------------------

* -> CPICXXXXXX, RFC_XXXXXX.TRC, CAD_RFC.LOG or DEV_RFC
* will be created.
* This parameter has no effect if you work with SAPRFC.INI
* In this file you can set the trace flag with parameter
* RFC_TRACE=1
CadRfcTrace                             0

* -> LOG_XXXX.LOG will be created
CadDialogStandardTrace                  0

* -> SAP_CAD.LOG will be created
CadDialogSapCadTrace                    1

* directory where all trace files should be written
CadTraceDir

*-----------------------------------------------------
* ABAP debug parameter   0 - set debug off
*                        1 - set debug on
*-----------------------------------------------------

* ABAP debug parameter
* If this parameter is set to 1 the ABAP-Debugger is started
* when the first SAP function is called
CadRfcAbapDebug                         0

*-------------------------------------------------------------
```

File cadrfc.ini

```
* Parameters needed if the connection is established from SAP
*-------------------------------------------------------------

* Directory for TEMP data (only necessary if you establish the
connection
* starting from SAP)
CadRfcTempDir                           c:\

* Router-Name (only necessary if you establish the connection
* starting from SAP)
CadRfcRouter

*----------------------------------------
* Default values for Dialog-RFC functions
*----------------------------------------

* Defaults BOM position (create and change)
DefaultCsrfcitemAusch
DefaultCsrfcitemBeikz
DefaultCsrfcitemErskz
DefaultCsrfcitemFmeng
DefaultCsrfcitemMeins
DefaultCsrfcitemPostp                  L
DefaultCsrfcitemPotx1
DefaultCsrfcitemPotx2
DefaultCsrfcitemRvrel
DefaultCsrfcitemSanfe
DefaultCsrfcitemSanin
DefaultCsrfcitemSanka
DefaultCsrfcitemSanko
DefaultCsrfcitemSanvs
DefaultCsrfcitemSchgt
DefaultCsrfcitemSortf
DefaultCsrfcitemStkkz
DefaultCsrfcitemStlal
```

```
DefaultCsrfcitemEwahr
DefaultCsrfcitemAlpgr
DefaultCsrfcitemDspst
DefaultCsrfcitemPrvbe
DefaultCsrfcitemNfeag
DefaultCsrfcitemNfgrp
DefaultCsrfcitemKzkup
DefaultCsrfcitemIntrm
DefaultCsrfcitemKzclb

* Defaults BOM sub item (create and change)
DefaultCssubitemUpmng
DefaultCssubitemUptxt

* Defaults BOM header (create and change)
DefaultBicskAennrCreate
DefaultBicskAennrChange
DefaultBicskBmein
DefaultBicskBmeng
DefaultBicskDatub
DefaultBicskDatuv
DefaultBicskEmeng
DefaultBicskExstl
DefaultBicskLabor
DefaultBicskLosbs
DefaultBicskLosvn
DefaultBicskStktx
DefaultBicskStlal
DefaultBicskStlan                       1
DefaultBicskStlbe
DefaultBicskStlst
DefaultBicskWerks
DefaultBicskZtext
DefaultBicskRevlv
DefaultBicskLabor
```

File cadrfc.ini

```
DefaultBicskStype

* Defaults BOM (display)
DefaultBomAction                        3
DefaultBomAlternative
DefaultBomApplication                   BEST
DefaultBomChangeNumber
DefaultBomRevisionLevel
DefaultBomType                          2

* Defaults DMS
DefaultDocDokar                         DRW
DefaultDocDoktl
DefaultDocDokvr
DefaultDocDoknrVl
DefaultDocDoktlVl
DefaultDocDokvrVl
DefaultDocDokst
DefaultDocProtf
DefaultDocLabor
DefaultDocAennr
DefaultDocPrenr
DefaultDocPretl
DefaultDocPrear
DefaultDocPrevr
DefaultDocBegru
DefaultDocDttrg
DefaultDocDappl                         ACD
DefaultDocDttrg1
DefaultDocDappl1
DefaultDocClassType                     017
DefaultDocClassNo

* Defaults DMS search
DefaultDocSearchDokar                   DRW
```

```
DefaultDocSearchUserDwnam
DefaultDocSearchAennr
DefaultDocSearchLabor
DefaultDocSearchDokst                  IA

* Defaults material master
DefaultMatChangeNumber
DefaultMatApplication                  M
DefaultMatType                         HALB
DefaultMatRefMaterial
DefaultMatClassType                    001
DefaultMatClassNo

* Defaults equipment master
DefaultEquClassType                    002
DefaultEquType                         M

* Defaults functional location
DefaultFloClassType                    003
DefaultFloStructIndicator              A
DefaultFloType                         C

* Defaults PM notifications
DefaultNotType                         M2

* Defaults PM orders
DefaultPmOrderType                     PM01

* General defaults
DefaultValidFrom
DefaultValidTo
DefaultPlant                           0001
DefaultBusinessArea                    0001

***********************************************************************
```

File cadrfc.ini

```
**                                                                  **
** You can specify default values for following parameters:         **
**                                                                  **
**    CadRfcUser            saptest  (User name)                    **
**    CadRfcPassword        demo     (Password - only for testing)  **
**    CadRfcClient          000      (SAP client)                   **
**    CadRfcLanguage        E        (Logon language)               **
**                                                                  **
** There are different ways how to connect to the SAP System:       **
**                                                                  **
** 1: use file SAPRFC.INI                                           **
** ----------------------                                           **
**    In file SAPRFC.INI you can specify the connection parameters. **
**    Entries in this file are mandatory if you want to connect to  **
**    different SAP systems without changing the configuration      **
**    parameters in CADRFC.INI.                                     **
**    For further information see file SAPRFC.INI.                  **
**    This file can be found on the SAP-GUI CD in directory /rfcsdk **
**                                                                  **
**    Configuration example:                                        **
**      CadRfcDestination                  ALR    (SAP system name) **
**      CadRfcHostname                            (Space)           **
**      CadRfcSysnr                               (Space)           **
**      CadRfcMessageServer                       (Space)           **
**                                                                  **
**    Configuration example SAPRFC.INI:                             **
**                                                                  **
**      Type A: R/3 system - specific application server            **
**      ----------------------------------------------              **
**      DEST=ALR       (SAP system name)                            **
**      TYPE=A         (connect to a specific R/3 application server)**
**      ASHOST=hs1026  (hostname of the application server)         **
**      SYSNR=86       (R/3 system number)                          **
**      RFC_TRACE=0    (RFC trace 0/1 : OFF/ON)                     **
**      ABAP_DEBUG=0   (ABAP debut 0/1 : OFF/ON)                    **
```

```
**      USE_SAPGUI=1   (use SAP-GUI 0/1 : OFF/ON)                    **
**                                                                    **
** 2: connect with load balance                                      **
** ----------------------------                                       **
**    connect to the SAP system using the load balance               **
**    functionality. There are two possibilities how to do this:     **
**                                                                    **
**    a) find message server automatically                            **
**    -----------------------------------                             **
**       The name of the message server is found automatically in    **
**       file SAPMSG.INI. This file is located in the Windows        **
**       directory (e.g. c:\winnt\sapmsg.ini). On NON-Windows
**
**       platforms you can copy the file to your working directory or**
**       set the environment "RFC_LOGON_INI_PATH"                    **
**                                                                    **
**       Configuration example:                                       **
**         CadRfcDestination                ALR    (SAP system name) **
**         CadRfcHostname                          (Space)           **
**         CadRfcSysnr                             (Space)           **
**         CadRfcUseSapMsgFile              Y      (use sapmsg.ini)  **
**         CadRfcMessageServer                     (Space)           **
**         CadRfcGroup                      PUBLIC (Logon group)     **
**                                                                    **
**    a) specify message server                                       **
**    -------------------------                                       **
**       It is possible to enter the message server name explicitly  **
**       You must set CadRfcUseSapMsgFile to N (no).                 **
**                                                                    **
**       Configuration example:                                       **
**         CadRfcDestination                ALR    (SAP system name) **
**         CadRfcHostname                          (Space)           **
**         CadRfcSysnr                             (Space)           **
**         CadRfcUseSapMsgFile              N      (no)              **
**         CadRfcMessageServer              us0311 (hostname)        **
```

File cadrfc.ini

```
**         CadRfcGroup                      PUBLIC (Logon group)     **
**                                                                   **
** 3: connect to a specified application server                      **
** -------------------------------------------                       **
**    You can enter the application server name and the system number**
**    of the SAP system you want to connect to.                      **
**                                                                   **
**       Configuration example:                                      **
**         CadRfcDestination                        (Space)           **
**         CadRfcHostname                   hs0026 (hostname)        **
**         CadRfcSysnr                      86     (system number)   **
**         CadRfcUseSapMsgFile              N      (no)              **
**         CadRfcMessageServer                     (Space)           **
**                                                                   **
** Attention: The file SIDEINFO is only used for R/2 connections.    **
**            For R/3 connection use file SAPRFC.INI.                **
**                                                                   **
** Connection to an R/2 system                                       **
** ---------------------------                                       **
**   If you want to connect to an R/2 system the parameter
**
**   CadDialogSapVersion must be set to R/2 and the connection       **
**   parameters must be defined in file SIDEINFO.                   **
**   You can only work with a CPIC user if you establish the         **
**   connection with an R/2 system.                                  **
**********************************************************************
```

File *caddialg.h*

```
/*-----------------------------------------------------------------*/
/*                                                                 */
/*  SAP AG Walldorf                                                */
/*  SAP  Systeme, Anwendungen und Produkte in der Datenverarbeitung  */
/*                                                                 */
/*  (C) Copyright SAP AG 1991                                      */
/*                                                                 */
/*-----------------------------------------------------------------*/
/*  File: caddialg.h                                               */
/*-----------------------------------------------------------------*/
/*  Headerfile for CAD dialog interface                            */
/*-----------------------------------------------------------------*/

#ifndef _CADDIALG_DEFINED
#define _CADDIALG_DEFINED

/******************* Structures ************************************/
typedef struct {
   char *pcode;
   char *ucode;
   int  subrc;
   int  meslen;
   long datalen;
   char fkz;
   long cpicrc;
   char *cpicmes;
} SAPHD;

/************** Function prototypes **********************************/

SAPHD    SapBomcn          (char*,char*,char**,char**);
SAPHD    SapBomcr          (char*,char*,char**,char**);
SAPHD    SapBomrq          (char*,char*,char**,char**);
```

File caddialg.h

```
SAPHD   SapBomwu        (char*,char*,char**,char**);
SAPHD   SapBreak        (char**);
SAPHD   SapChgch        (char*,char*,char**,char**);
SAPHD   SapChgcr        (char*,char*,char**,char**);
SAPHD   SapChgrq        (char*,char*,char**,char**);
SAPHD   SapConnc        (char**,char*,char*,char*,char*,
                char*,char*,char*,char**);
SAPHD   SapConncRfc     (char*,char*, char**,char**);
SAPHD   SapDebug        (char*);
SAPHD   SapDisco        ();
SAPHD   SapDocch        (char*,char*,char**,char**);
SAPHD   SapDocco        (char*,char*,char**,char**);
SAPHD   SapDocci        (char*,char*,char**,char**);
SAPHD   SapDoccr        (char*,char*,char**,char**);
SAPHD   SapDocrq        (char*,char*,char**,char**);
SAPHD   SapEcrch        (char*,char*,char**,char**);
SAPHD   SapEcrcr        (char*,char*,char**,char**);
SAPHD   SapEcrrq        (char*,char*,char**,char**);
SAPHD   SapEqich        (char*,char*,char**,char**);
SAPHD   SapEqicr        (char*,char*,char**,char**);
SAPHD   SapEqiex        (char*,char*,char**,char**);
SAPHD   SapEqirq        (char*,char*,char**,char**);
SAPHD   SapHlprq        (char*,char*,char**,char**);
SAPHD   SapMatch        (char*,char*,char**,char**);
SAPHD   SapMatcr        (char*,char*,char**,char**);
SAPHD   SapMatrq        (char*,char*,char**,char**);
SAPHD   SapMtnce        (char*,char*,char**,char**);
SAPHD   SapPmncr        (char*,char*,char**,char**);
SAPHD   SapPmnch        (char*,char*,char**,char**);
SAPHD   SapPmnrq        (char*,char*,char**,char**);
SAPHD   SapRsrve        (char*,char*,char**,char**);
SAPHD   SapSlcls        (char*,char*,char**,char**);
SAPHD   SapSlobj        (char*,char*,char**,char**);
SAPHD   SapSmail        (char*,char*,char**,char**);
SAPHD   SapSysrq        (char*,char*,char**,char**);
```

```
SAPHD   SapTpcrq        (char*,char*,char**,char**);
SAPHD   SapUsr01        (char*,char*,char**,char**);
SAPHD   SapUsr02        (char*,char*,char**,char**);
SAPHD   SapUsr03        (char*,char*,char**,char**);
SAPHD   SapUsr04        (char*,char*,char**,char**);
SAPHD   SapUsr05        (char*,char*,char**,char**);
SAPHD   SapVersi        (char**);

#endif /* _CADDIALG_DEFINED */
```

File *cadrfc.h*

```
/*@(#) cadrfc.h      20.13     SAP     %E */
/*-----------------------------------------------------------------*/
/*                                                                 */
/*  SAP AG Walldorf                                                */
/*  SAP  Systeme, Anwendungen und Produkte in der Datenverarbeitung  */
/*                                                                 */
/*  (C) Copyright SAP AG 1995                                      */
/*                                                                 */
/*-----------------------------------------------------------------*/
/*  File: cadrfc.h                                                 */
/*-----------------------------------------------------------------*/
/*  global headerfile for CAD RFC interface                        */
/*-----------------------------------------------------------------*/
/*  4.0A                                                           */
/*  BINK123657 080997 new function CadRfcWriteFieldTabEntry        */
/*                    new function CadRfcWriteCharTabEntry         */
/*  BINK121255 110897 new structure CSSUBITEM                      */
/*                    new function CadRfcGetLastError              */
/*                    new function CadRfcGetCharTabEntry           */
/*                    new function CadBomChangeSubItem             */
/*                    new function CadBomChangeSubItemRead         */
/*                    new function CadBomCreateSubItem             */
/*                    new function CadBomDisplaySubItem            */
/*                    new function CadRfcGetBomItemEntry           */
/*                    new function CadRfcGetBomSubItemEntry        */
/*                    new function CadRfcWriteBomSubItemEntry      */
/*  BINK117409 050897 BICSK for 3.X and 4.0                        */
/*             110797 new function CadRfcGetEquipmentData          */
/*                    new function CadRfcGetMaterialData           */
/*                    new function CadRfcGetFuncLocdata            */
/*                    new fields in Import-Structure               */
/*  BINK110955 280497 new function CadRfcGetFieldTabEntry          */
/*                    new parameter for matchcode selection        */
```

```
/*          150597 get rid of compiler warnings                  */
/*  3.1G                                                        */
/*  BINK104200 280197 new fields in CSRFCITEM                   */
/*  3.0E                                                        */
/*  BINK090600 180696 new fields in CSRFCITEM                   */
/*-------------------------------------------------------------------*/

#include "saprfc.h"
#include "sapitab.h"

/******* field type definitions and structure type definitions *******/
#ifndef SAP_ST_SAP_FIELD_DATA
#define SAP_ST_SAP_FIELD_DATA
typedef struct
  {
  RFC_CHAR Fieldmulti[40];
  RFC_CHAR Fieldname[35];
  RFC_CHAR Fieldvalue[255];
  } SAP_FIELD_DATA;
#endif

#ifndef SAP_ST_CLASS_CHARACTERISTICS
#define SAP_ST_CLASS_CHARACTERISTICS
typedef struct
  {
  RFC_CHAR Atnam[30];
  RFC_CHAR Atbez[30];
  RFC_CHAR Atwrt[30];
  RFC_CHAR Dinkb[3];
  RFC_CHAR Unit[6];
  } CLASS_CHARACTERISTICS;
#endif

#ifndef SAP_ST_BICSK
#define SAP_ST_BICSK
```

File cadrfc.h

```
typedef struct
 {
 RFC_CHAR Stype[1];
 RFC_CHAR Tcode[4];
 RFC_CHAR Aennr[12];
 RFC_CHAR Bmein[3];
 RFC_CHAR Bmeng[17];
 RFC_CHAR Cadkz[1];
 RFC_CHAR Datub[10];
 RFC_CHAR Datuv[10];
 RFC_CHAR Emeng[17];
 RFC_CHAR Equnr[18];
 RFC_CHAR Exstl[18];
 RFC_CHAR Labor[3];
 RFC_CHAR Loekz[1];
 RFC_CHAR Losbs[17];
 RFC_CHAR Losvn[17];
 RFC_CHAR Matnr[18];
 RFC_CHAR Selal[2];
 RFC_CHAR Serge[30];
 RFC_CHAR Stktx[40];
 RFC_CHAR Stlal[2];
 RFC_CHAR Stlan[1];
 RFC_CHAR Stlbe[4];
 RFC_CHAR Stlst[2];
 RFC_CHAR Vmtnr[18];
 RFC_CHAR Werks[4];
 RFC_CHAR Ztext[40];
 RFC_CHAR Revlv[2];
 RFC_CHAR Tplnr[30];
 RFC_CHAR Dokar[3];
 RFC_CHAR Doknr[25];
 RFC_CHAR Doktl[3];
 RFC_CHAR Dokvr[2];
 RFC_CHAR Vbeln[10];
```

```
  RFC_CHAR Vbpos[6];
  RFC_CHAR Stobj[18];
  RFC_CHAR Vdknr[25];                    /* 050897 */
  RFC_CHAR Vdkar[3];                     /* 050897 */
  RFC_CHAR Vdktl[3];                     /* 050897 */
  RFC_CHAR Vdkvr[2];                     /* 050897 */
  RFC_CHAR Veqnr[18];                    /* 050897 */
  RFC_CHAR Vtpnr[30];                    /* 050897 */
  RFC_CHAR Tcode2[20];                   /* 050897 */
  } BICSK;
#endif

#ifndef SAP_ST_CSRFCITEM
#define SAP_ST_CSRFCITEM
typedef struct
  {
  RFC_CHAR Upskz[1];
  RFC_CHAR Auskz[1];
  RFC_CHAR Alpos[1];
  RFC_CHAR Ausch[6];
  RFC_CHAR Avoau[6];
  RFC_CHAR Beikz[1];
  RFC_CHAR Cadpo[1];
  RFC_CHAR Ekgrp[3];
  RFC_CHAR Erskz[1];
  RFC_CHAR Fmeng[1];
  RFC_CHAR Idnrk[18];
  RFC_CHAR Lifnr[10];
  RFC_CHAR Lifzt[3];
  RFC_CHAR Matkl[9];
  RFC_CHAR Meins[3];
  RFC_CHAR Menge[18];
  RFC_CHAR Netau[1];
  RFC_CHAR Nfmat[18];
  RFC_CHAR Nlfzt[3];
```

```
RFC_CHAR Peinh[6];
RFC_CHAR Posnr[4];
RFC_CHAR Postp[1];
RFC_CHAR Preis[14];
RFC_CHAR Potx1[40];
RFC_CHAR Potx2[40];
RFC_CHAR Pswrk[4];
RFC_CHAR Rekrs[1];
RFC_CHAR Rform[2];
RFC_CHAR Roanz[17];
RFC_CHAR Roame[3];
RFC_CHAR Rokme[3];
RFC_CHAR Romei[3];
RFC_CHAR Romen[17];
RFC_CHAR Roms1[17];
RFC_CHAR Roms2[17];
RFC_CHAR Roms3[17];
RFC_CHAR Rvrel[1];
RFC_CHAR Sakto[10];
RFC_CHAR Sanfe[1];
RFC_CHAR Sanin[1];
RFC_CHAR Sanka[1];
RFC_CHAR Sanko[1];
RFC_CHAR Sanvs[1];
RFC_CHAR Schgt[1];
RFC_CHAR Selal[2];
RFC_CHAR Selid[18];
RFC_CHAR Selpo[4];
RFC_CHAR Selsb[10];
RFC_CHAR Sortf[10];
RFC_CHAR Stkkz[1];
RFC_CHAR Stktx[40];
RFC_CHAR Stlal[2];
RFC_CHAR Stlkz[1];
RFC_CHAR Verti[4];
```

```
  RFC_CHAR Webaz[3];
  RFC_CHAR Waers[5];
  RFC_CHAR Dokar[3];
  RFC_CHAR Doknr[25];
  RFC_CHAR Dokvr[2];
  RFC_CHAR Doktl[3];
  RFC_CHAR Ewahr[3];
  RFC_CHAR Ekorg[4];
  RFC_CHAR Lgort[4];
  RFC_CHAR Class[18];
  RFC_CHAR Klart[3];
  RFC_CHAR Potpr[1];
  RFC_CHAR Alpgr[2];
  RFC_CHAR Alpst[1];
  RFC_CHAR Alprf[2];
  RFC_CHAR Dspst[2];          /* 180696 */
  RFC_CHAR Prvbe[10];          /* 180696 */
  RFC_CHAR Nfeag[2];          /* 180696 */
  RFC_CHAR Nfgrp[2];          /* 180696 */
  RFC_CHAR Kzkup[1];          /* 180696 */
  RFC_CHAR Intrm[18];         /* 280197 */
  RFC_CHAR Kzclb[1];          /* 280197 */
} CSRFCITEM;
#endif

#ifndef SAP_ST_CSSUBITEM          /* 110897 */
#define SAP_ST_CSSUBITEM
typedef struct
{
  RFC_CHAR Posid[4];
  RFC_CHAR Ebort[20];
  RFC_CHAR Upmng[17];
  RFC_CHAR Uposz[4];
  RFC_CHAR Uptxt[40];
} CSSUBITEM;
```

File cadrfc.h

```
#endif

#ifndef SAP_ST_DRAW_RFC
#define SAP_ST_DRAW_RFC
typedef struct {
  RFC_CHAR Dokar[3];
  RFC_CHAR Doknr[25];
  RFC_CHAR Dokvr[2];
  RFC_CHAR Doktl[3];
  RFC_CHAR Dwnam[12];
  RFC_CHAR Dokst[2];
  RFC_CHAR Protf[20];
  RFC_CHAR Labor[3];
  RFC_CHAR Aennr[12];
  RFC_CHAR Werka[4];
  RFC_CHAR Loedk[1];
  RFC_CHAR Cadkz[1];
  RFC_CHAR Prenr[25];
  RFC_CHAR Prevr[2];
  RFC_CHAR Pretl[3];
  RFC_CHAR Prear[3];
  RFC_CHAR Begru[4];
  RFC_CHAR Filep[70];
  RFC_CHAR Dttrg[10];
  RFC_CHAR Dappl[3];
  RFC_CHAR Filep1[70];
  RFC_CHAR Dttrg1[10];
  RFC_CHAR Dappl1[3];
  RFC_CHAR Vrldat[10];
  RFC_NUM  Vprior[1];
  RFC_CHAR Res1[14];
  RFC_CHAR Res2[14];
  RFC_CHAR Res3[14];
  RFC_CHAR Res4[14];
  RFC_CHAR Adatum[10];
```

```
  RFC_CHAR DoknrVl[25];
  RFC_CHAR DoktlVl[3];
  RFC_CHAR DokvrVl[2];
  RFC_CHAR Doktxt[40];
  RFC_CHAR Pathname[200];
  } DRAW_RFC;
#endif

typedef struct {
   char BomAction[1];
   char BomAlternative[2];
   char BomApplication[4];
   char BomType[1];
   char BusinessArea[4];
   char ChangeNumber[12];
   char ClassNo[18];
   char ClassType[3];
   char DocNumber[25];
   char DocOriginal[70];
   char Doktl[3];
   char DocPathname[70];
   char DocPathprefix[10];
   char Dokar[3];
   char Dokvr[2];
   char Dokst[2];
   char Dwnam[12];
   char Equipment[18];
   char EquipmentType[1];
   char FunctionalLocation[30];
   char FunctionalLocationType[1];
   char Labor[3];
   char MatApplication[1];
   char MatType[4];
   char Material[18];
   char Notification[12];
```

File cadrfc.h

```
  char NotificationType[2];
  char Object[1];
  char ObjectId[1];
  char ObjectKey[50];
  char ObjectTable[10];
  char PmOrderNo[12];
  char PmOrderType[4];
  char Plant[4];
  char PrevDocNumber[25];
  char PrevDocType[3];
  char PrevDocPart[3];
  char PrevDocVers[2];
  char RevisionLevel[2];
  char RefEquipment[18];
  char RefFunctionalLocation[30];
  char RefReferenceLocation[30];
  char RefMaterial[18];
  char SkipFirstScreen[1];
  char StructureIndicator[5];
  char ValidFrom[10];
  char ValidTo[10];
  char WUDirect[1];
  char WUViaclass[1];
  char ReturnMcdValues[1];          /* 280497 */
} CadRfcExportField;


typedef struct {
  char ChangeNumber[12];
  char ClassNo[18];
  char ClassType[3];
  char DateFormat[30];
  char DecimalSign[1];
  char DocNumber[25];
  char DocType[3];
```

```
  char DocPart[3];
  char DocVers[2];
  char DocPath[255];
  char Equipment[18];
  char FunctionalLocation[30];
  char FunctionCode[14];
  char Language[1];
  char Material[18];
  char Notification[12];
  char ObjectKey[50];
  char ObjectTable[10];
  char PmOrderNo[12];
  char SapSystemRelease[4];
  char UserName[12];
  char ValidFrom[10];
  char Message[200];              /* 110797 */
  char MessageLength[4];            /* 180797 */
} CadRfcImportField;


typedef struct {
  char ApplicationType[11];
  char ApplicationMode;
  char ApplicationName[41];
  char ApplicationPath[71];
  char TempDir[141];
  char CallbackPara0[133];
  char CallbackPara1[40];
  char CallbackPara2[40];
  char CallbackPara3[40];
  char CallbackPara4[40];
} CadRfcInitData;

CadRfcInitData *pInitData;
```

File cadrfc.h

```
/******************* Function prototypes ***************************/

int  CadBomChangeSubItemRead (BICSK *, CadRfcImportField *, /* 110897 */
                        ITAB_H **, ITAB_H **);
int  CadBomChangeSubItem     (BICSK *, CadRfcImportField *, /* 110897 */
                        ITAB_H **, ITAB_H **);
int  CadBomCreateSubItem     (BICSK *, CadRfcImportField *, /* 110897 */
                        ITAB_H **, ITAB_H **);
int  CadBomDisplaySubItem    (BICSK *, CadRfcImportField *, /* 110897 */
                        ITAB_H **, ITAB_H **);
int  CadRfcAllocMemory       (CadRfcExportField **, CadRfcImportField **,
                        BICSK **, DRAW_RFC **);
int  CadRfcBomChange         (BICSK *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcBomCreate         (BICSK *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcBomDisplay        (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcBomWhereUsed      (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcClassSelection    (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **, ITAB_H **);
int  CadRfcClose             (void);                    /* 150597 */

int  CadRfcChgChange         (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcChgCreate         (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcChgDisplay        (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcDistributeApplication(int, char**);

int  CadRfcDocChange         (DRAW_RFC *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
```

```
int  CadRfcDocCreate        (DRAW_RFC *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcDocDisplay       (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcDocSelect        (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcDocInsert        (DRAW_RFC *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcGetBomItemEntry   (int, CSRFCITEM *);                    /* 110897 */

int  CadRfcGetBomSubItemEntry(int, CSSUBITEM *);                    /* 110897 */

int  CadRfcGetCharTabEntry   (int, ITAB_H *, CLASS_CHARACTERISTICS *);  /* 110897 */

int  CadRfcGetDmsData       (CadRfcExportField *, DRAW_RFC *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcGetEquipmentData  (CadRfcExportField *, CadRfcImportField *, /* 110797 */
                        ITAB_H **, ITAB_H **);
int  CadRfcGetFuncLocData    (CadRfcExportField *, CadRfcImportField *, /* 110797 */
                        ITAB_H **, ITAB_H **);
int  CadRfcGetMaterialData   (CadRfcExportField *, CadRfcImportField *, /* 110797 */
                        ITAB_H **, ITAB_H **);
int  CadRfcGetFieldTabEntry  (int, ITAB_H *, SAP_FIELD_DATA *);        /* 280497 */

int  CadRfcGetLastError      (char *);                          /* 110897 */

int  CadRfcGetHandle        (RFC_HANDLE*);

int  CadRfcGetSystemParas    (CadRfcImportField *);

int  CadRfcSetHandle        (RFC_HANDLE*);

int  CadRfcEquChange        (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcEquCreate        (CadRfcExportField *, CadRfcImportField *,
```

File cadrfc.h

```
                    ITAB_H **, ITAB_H **);
int  CadRfcEquDisplay       (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcFloChange        (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcFloCreate        (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcFloDisplay       (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcPmOrderChange    (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcPmOrderCreate    (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcPmOrderDisplay   (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcPmrChange        (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcPmrCreate        (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcPmrDisplay       (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcMailInbox        (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcMailOutbox       (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcWorkflowInbox    (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcMatchcode        (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcMatChange        (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcMatCreate        (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
int  CadRfcMatDisplay       (CadRfcExportField *, CadRfcImportField *,
                    ITAB_H **, ITAB_H **);
```

```
int  CadRfcProductStructure  (CadRfcExportField *, CadRfcImportField *,
                        ITAB_H **, ITAB_H **);
int  CadRfcInit             (char *, char*, char*, char*, RFC_HANDLE*);

int  CadRfcOpen             (char *, char*, char*, char*, char*, char*,
                        char *, RFC_HANDLE*);
int  CadRfcWriteBomTabEntry  (char*, char*, int);

int  CadRfcWriteBomSubItemEntry (char*, char*, int);          /* 110897 */

int  CadRfcWriteCharTabEntry (char*, char*, int);             /* 080997 */

int  CadRfcWriteFieldTabEntry(char*, char*, int);             /* 080997 */
```

File *cadtst1.c*

```
/*-----------------------------------------------------------------*/
/*                                                                 */
/*  SAP AG Walldorf                                                */
/*  SAP  Systeme, Anwendungen und Produkte in der Datenverarbeitung  */
/*                                                                 */
/*  (C) Copyright SAP AG 1992                                      */
/*                                                                 */
/*-----------------------------------------------------------------*/
/*  File: cadtst1.c                                                */
/*-----------------------------------------------------------------*/
/*  Simple test program for the CAD dialog interface               */
/*  Program connects to SAP and requests material master           */
/*  informations.                                                  */
/*                                                                 */
/*  Routines:                                                      */
/*  ---------                                                      */
/*  main()          main program                                   */
/*-----------------------------------------------------------------*/
/*  1.10                                                           */
/*  YENK000000 290693 Aenderungen fuer DOS/Windows                 */
/*  1.00                                                           */
/*  YENK000000 210892 Neu aufgenommen                              */
/*-----------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "caddialg.h"

char    sndstr[16500];



/*-----------------------------------------------------------------*/
```

File cadtst1.c

```
/* main                                                    */
/*-------------------------------------------------------------------*/
/*          NAME          DESCRIPTION                          */
/*          --------------- ----------------------------------  */
/*   INPUT :   argc                                          */
/*          argv                                            */
/*   OUTPUT:   ----                                          */
/*-------------------------------------------------------------------*/
/*   RETURN:   ----                                          */
/*-------------------------------------------------------------------*/
main (argc,argv)
int argc;
char *argv[];
{ SAPHD    hd;
  char     *rcvstr;
  char     *sapmes;
  char     *version;

/* Connect: Parameters about SAP system, user name and password      */
/* will be read in configuration file caddialg.ini.                  */

  hd=SapVersi(&version);
  printf("SAP-CAD function library, interface version: %s\n",version);
  printf("-----------------------------------------------\n");
  printf("\n");

  hd = SapConnc(argv,"","","","","","","",&sapmes);

  printf("CONNC --- CPIC-RC=%ld\n",hd.cpicrc);
  printf("--------------------\n");
  if (hd.cpicrc == 0)
    printf("Received Message: %s\n\n",sapmes);
  else {
    printf("Received CPIC-Error: %s\n\n",hd.cpicmes);
    exit(1);
```

File cadtst1.c

```
  }

/* Request material master. All fields for send and receive          */
/* must be defined in the SAP customizing tables TCIM, TCIU, and     */
/* TCID.                                                             */

  strcpy(sndstr,"000047110101");
  hd = SapMatrq("MR",sndstr,&rcvstr,&sapmes);

  printf("MATRQ --- CPIC-RC=%ld\n",hd.cpicrc);
  printf("--------------------\n");
  if (hd.cpicrc == 0) {
    printf("Sended Data:   %s\n",sndstr);
    rcvstr[hd.datalen] = 0x0;
    printf("Received Data: %s\n",rcvstr);
    sapmes[hd.meslen] = 0x0;
    printf("Received Message: %s\n\n",sapmes);
  }
  else {
    printf("Received CPIC-Error: %s\n\n",hd.cpicmes);
    exit(1);
  }

/* Disconnect                                                        */

  hd = SapDisco();
  printf("DISCO --- CPIC-RC=%ld\n",hd.cpicrc);
  return(0);
}
```