

APIs in der Logistik



Release 4.6C



Copyright

© Copyright 2001 SAP AG. Alle Rechte vorbehalten.

Weitergabe und Vervielfältigung dieser Publikation oder von Teilen daraus sind, zu welchem Zweck und in welcher Form auch immer, ohne die ausdrückliche schriftliche Genehmigung durch SAP AG nicht gestattet. In dieser Publikation enthaltene Informationen können ohne vorherige Ankündigung geändert werden.

Die von SAP AG oder deren Vertriebsfirmen angebotenen Software-Produkte können Software-Komponenten auch anderer Software-Hersteller enthalten.

Microsoft[®], WINDOWS[®], NT[®], EXCEL[®], Word[®], PowerPoint[®] und SQL Server[®] sind eingetragene Marken der Microsoft Corporation.

IBM[®], DB2[®], OS/2[®], DB2/6000[®], Parallel Sysplex[®], MVS/ESA[®], RS/6000[®], AIX[®], S/390[®], AS/400[®], OS/390[®] und OS/400[®] sind eingetragene Marken der IBM Corporation.

ORACLE[®] ist eine eingetragene Marke der ORACLE Corporation.

INFORMIX[®]-OnLine for SAP und Informix[®] Dynamic Server[™] sind eingetragene Marken der Informix Software Incorporated.

UNIX[®], X/Open[®], OSF/1[®] und Motif[®] sind eingetragene Marken der Open Group.




HTML, DHTML, XML, XHTML sind Marken oder eingetragene Marken des W3C[®], World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA[®] ist eine eingetragene Marke der Sun Microsystems, Inc.

JAVASCRIPT[®] ist eine eingetragene Marke der Sun Microsystems, Inc., verwendet unter der Lizenz der von Netscape entwickelten und implementierten Technologie.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo und mySAP.com sind Marken oder eingetragene Marken der SAP AG in Deutschland und vielen anderen Ländern weltweit. Alle anderen Produkte sind Marken oder eingetragene Marken der jeweiligen Firmen.

Symbole

Symbol	Bedeutung
	Achtung
	Beispiel
	Hinweis
	Empfehlung
	Syntax

Inhalt

APIs in der Logistik	5
Arbeiten mit APIs	6
Mögliche Szenarios	7
Liste der vorhandenen Funktionsgruppen	10
Liste der vorhandenen APIs	13
Allgemeine Abarbeitungsregeln	20
Systemmeldungen	22
Nomenklatur für die APIs	23
Identifikation der einzelnen Objekte.....	24
API Objekte	25
API Aktionen	26
Zuordnung von Beziehungswissen.....	28
Generierung von ABAP Coding	31
Allgemeine APIs (CALO).....	33
Initialisierung der APIs	34
APIs für die Protokollsteuerung	36

APIs in der Logistik

Einsatzmöglichkeiten

Application Programming Interfaces (APIs) sind Programmschnittstellen zu Anwendungen im R/3-System. Mit Hilfe von APIs können Sie von einem externen System Daten an das R/3-System übergeben und umgekehrt. Innerhalb des R/3-Systems können Daten von einer Anwendung an eine andere Anwendung übergeben werden, ohne daß dafür ein Dialog notwendig ist.

Funktionsumfang

Die APIs in folgenden Bereichen der Logistik werden behandelt:

CA-Klassensystem

CA-Merkmale

LO-Variantenkonfiguration

PP-Stücklisten

LO-Änderungsdienst

Arbeiten mit APIs

Arbeiten mit APIs

Funktionsumfang

APIs sollten grundsätzlich einfach in der Anwendung sein. Dazu sollten die folgenden Punkte erfüllt sein:

- einfache und überschaubare Datenübergabe; einheitliche Schnittstellen
- einheitlicher Aufbau der APIs
- einfacher Zugriff auf die APIs durch eigene Funktionsgruppen
- klar definierte und abgegrenzte Funktionalität
- für bestimmte Bereiche werden Funktionsmakros gebildet.

Die APIs werden auf verschiedenen Ebenen gebraucht. Viele Dialogfunktionen werden stufenweise als APIs zur Verfügung gestellt. Es gibt beispielsweise APIs:

- um auf (alle) vorhandenen Daten eines Objekts lesend zuzugreifen;
- um Daten (z. B. Merkmale, Merkmalwerte) in vorhandene Strukturen (Klassen, Tabellen etc.) einzupflegen;
- um neue Strukturen zu erzeugen, z. B. Änderungsstammsätze, Merkmale, Klassen, Konfigurationsprofile, Stücklisten, Beziehungswissen;
- für die Klassifikation von Objekten;
- für die Suche im Klassensystem;
- für die Konfiguration;
- um auf Basisdaten (Customizing) zuzugreifen (zur Zeit noch nicht verfügbar).

Die Funktionalität der APIs soll interne und externe Anforderungen an Schnittstellen gerecht werden. Der Zugriff muß deshalb sowohl über ABAP/4- Routinen als auch über externe Programme (über Remote Function Call) erfolgen können. Die APIs müssen deshalb auf DDIC - Strukturen aufbauen (RFC Fähigkeit).



Die ausführliche Dokumentation für die Anwendung der einzelnen APIs erhalten Sie in der Funktionsbausteindoku, die online im R/3 verfügbar ist.

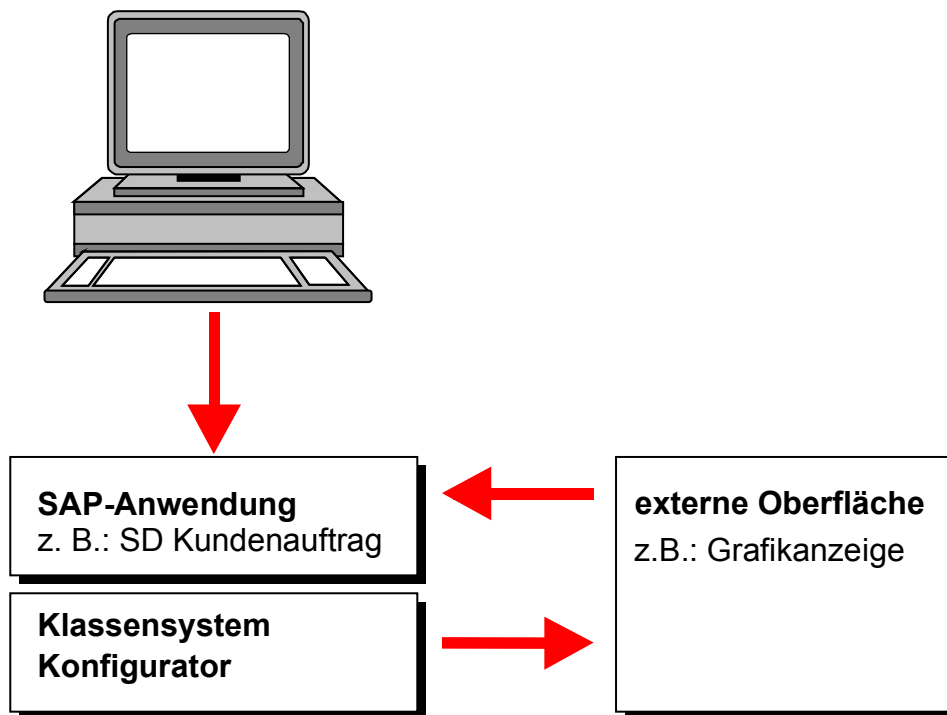
Mögliche Szenarios

Mit Hilfe von Application Programming Interfaces (APIs) können verschiedene Szenarios unterstützt werden. Dabei unterscheidet man, ob das R/3-System oder das externe System führend ist.

R/3-System ist führend

1. Das externe System wird aus der R/3-Anwendung aktiviert.

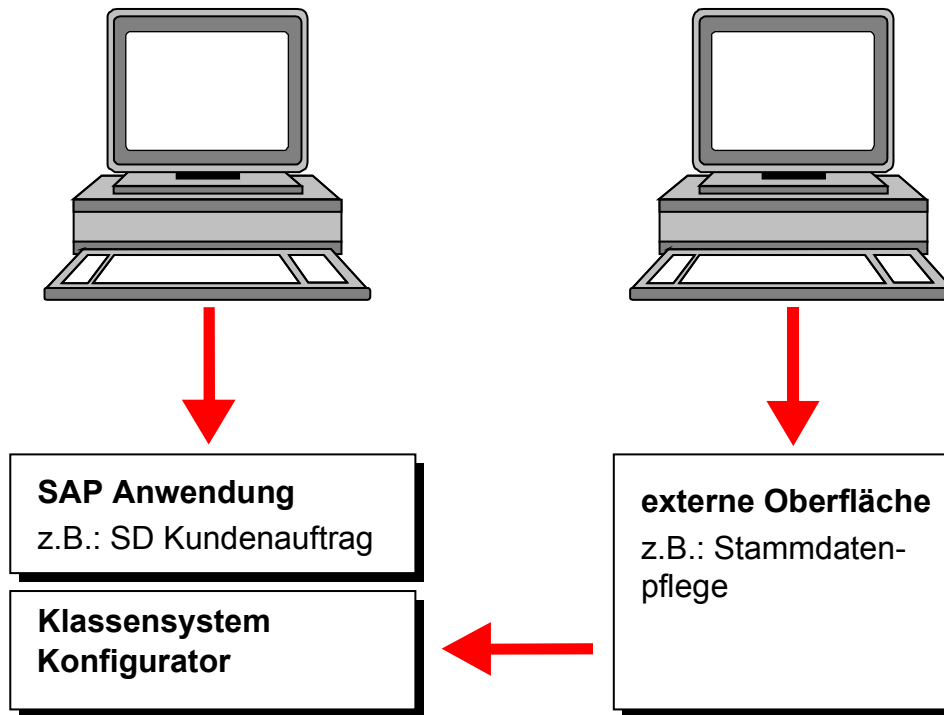
Dies ist zum Beispiel dann der Fall, wenn bei einer Konfiguration das Ergebnis grafisch angezeigt werden soll. Über APIs werden die Daten aus der Konfiguration an das externe System übergeben. Siehe folgende Grafik.



2. Datenimport per API

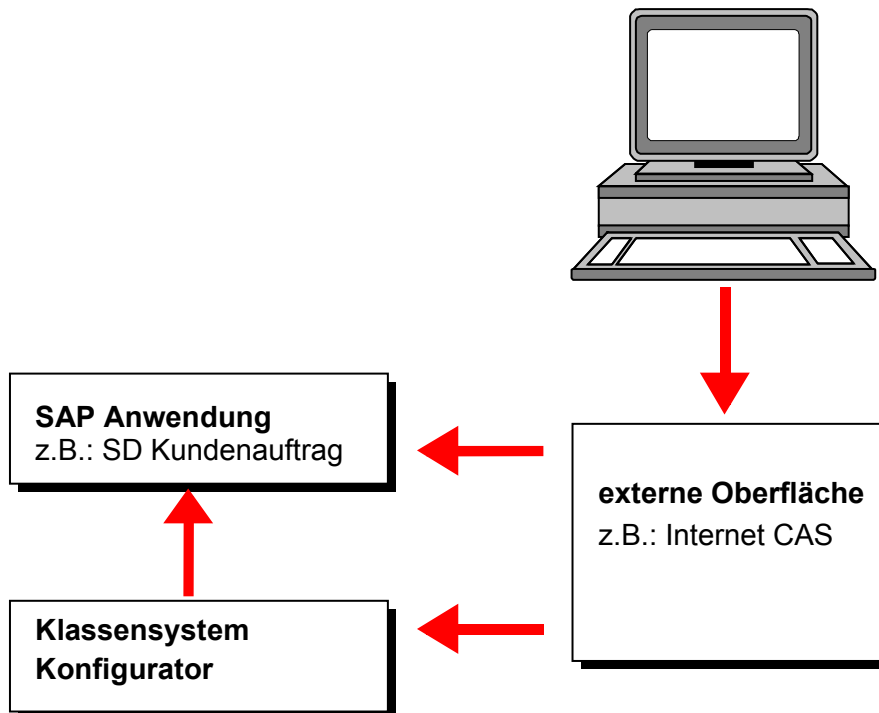
Die Stammdatenpflege erfolgt in einem externen System. Die Daten werden regelmäßig über APIs an das R/3-Klassensystem oder die Variantenkonfiguration übergeben. Siehe folgende Grafik.

Mögliche Szenarios



Externes System ist führend

Wenn das externe System führend ist, kann man über APIs auf R/3-Stammdaten und Funktionen wie z. B. die Klassifikation und den Konfigurator zugreifen. Das externe System läuft als unabhängige Anwendung oder als unabhängige Oberfläche: siehe folgende Grafik.



Beispielanwendungen:

- Über ein externes Sales System, das über Internet Produkte anbietet, können Sie z. B. mit dem R/3-Konfigurator ein Produkt konfigurieren
- Aus einem CAD-System heraus können im R/3-System Merkmale und Klassen angelegt und Materialien klassifiziert werden.

Liste der vorhandenen Funktionsgruppen

Liste der vorhandenen Funktionsgruppen

Entsprechend ihrer Funktion werden die APIs zu Gruppen zusammengefaßt. Darüber hinaus wird unterschieden zwischen:

- Allgemeinen APIs
- APIs für die Variantenkonfiguration und Klassifikation
- APIs für Stücklisten und Änderungsdienst

Allgemeine APIs

Gruppe: Protokoll

Funktionsgruppe CALO

(APIs for Logging)

Als anwendungsübergreifende APIs stehen Ihnen die Funktionsbausteine dieser Gruppe zum Protokollieren der API Aufrufe zur Verfügung. Mit diesen APIs können Sie das Protokollverfahren parametrisieren, das Protokoll lesen und gezielt schreiben.

APIs für die Variantenkonfiguration und Klassifikation

Die APIs für die Variantenkonfiguration und Klassifikation werden in der Entwicklungsklasse CL angelegt. Es existieren folgende Gruppen:

Gruppe: Daten der Variantenkonfiguration lesen

Funktionsgruppe CARD

(Configurator APIs for Reading Data)

Die APIs dieser Gruppe bieten die Funktionalität, auf die Daten der Variantenkonfiguration, z. B. Beziehungswissen und Tabellen direkt lesend zuzugreifen.

Gruppe: Merkmaldaten lesen

Funktionsgruppe CACTR

Die APIs dieser Gruppe ermöglichen das Lesen der Merkmaldaten wie z. B. Kennzeichen, Werte und zugeordnetes Beziehungswissen.

Gruppe: Klassendaten lesen

Funktionsgruppe CACLR

Die APIs dieser Gruppe ermöglichen das Lesen der Klassendaten wie z. B. Merkmale einer Klasse, Überschreibungen.

Gruppe Selektion

Funktionsgruppe CASE

(Object Selection)

Die APIs dieser Gruppe dienen zum Suchen von klassifizierten Objekten.
(Zur Zeit noch nicht verfügbar)

Gruppe: Klassifikation

Liste der vorhandenen Funktionsgruppen

Funktionsgruppe CACL

(Classification)

Die APIs dieser Gruppe ermöglichen das Anlegen neuer und das Ändern vorhandener Objektzuordnungen und ihrer Bewertung.



Die Funktionsgruppe CACL darf im Bereich "Beziehungswissen" nicht aufgerufen werden.

Gruppe: Datenpflege für die Variantenkonfiguration

Funktionsgruppe CAMA

(Maintenance)

Die APIs dieser Gruppe ermöglichen das Anlegen und Ändern von Stammdaten der Variantenkonfiguration, wie z. B. Konfigurationsprofil, Tabellen, Beziehungswissen etc.

Gruppe: Merkmaldaten pflegen

Funktionsgruppe CACTM

Die APIs dieser Gruppe ermöglichen die Pflege von Merkmaldaten wie z. B. Anlegen, Löschen, Ändern, Zuordnung von Beziehungswissen etc.

Gruppe: Klassendaten pflegen

Funktionsgruppe CACLM

Die APIs dieser Gruppe ermöglichen die Pflege von Klassendaten wie z. B. Anlegen, Ändern, Löschen, Zuordnung von Beziehungswissen zu den Merkmalen und Merkmalwerten etc.



Die APIs der folgenden Funktionsgruppen CAVC_... lösen die APIs der Funktionsgruppe CACO ab.

Gruppe: Initialisierung und Abschluß einer Konfiguration

Funktionsgruppe CAVC_OBJECTS

Die APIs dieser Gruppe ermöglichen die Initialisierung und das Beenden der Variantenkonfiguration. Das heißt, sie bearbeiten die Objekte auf die die Variantenkonfiguration angewendet werden kann (z.B. Auftragsposition).

Gruppe: Behandlung der kompletten Konfiguration

Funktionsgruppe CAVC_CFG

Die APIs dieser Gruppe behandeln die Konfiguration als Ganzes (z.B. Stücklistenauflösung oder Liste aller Instanzen).

Gruppe: Behandlung von einzelnen Instanzen

Funktionsgruppe CAVC_INST

Die APIs dieser Gruppe behandeln nur eine einzelne Instanz. Alle APIs dieser Gruppe besitzen die Instanznummer als Eingabeparameter.

Gruppe: Dialogfunktionen

Liste der vorhandenen Funktionsgruppen

Funktionsgruppe CAVC_DIALOG

Die APIs, die den SAP-GUI verwenden, um einen Dialog zu verarbeiten, werden in dieser Gruppe gesammelt.

APIs für Stücklisten und Änderungsdienst

Alle APIs werden in den Entwicklungsklassen CS (Stücklisten) und CC (Änderungsdienst) angelegt. Die Aufteilung der Funktionsgruppen erfolgt folgendermaßen:

Gruppe: Änderungsstammdaten lesen und pflegen

Funktionsgruppe CCAP

Die APIs dieser Gruppe ermöglichen das Lesen, Anlegen und Ändern von Änderungsstammdaten.

Gruppe: Stücklistendaten lesen und pflegen

Funktionsgruppe CSAP

Die APIs dieser Gruppe ermöglichen das Lesen, Anlegen und Ändern von Stücklistendaten.

Liste der vorhandenen APIs

CALO

Gruppe: Protokoll

CALO_COLLECT_MESSAGES_ON_OFF	Meldungen werden intern gesammelt
CALO_DB_LOG_ON_OFF	DB-logging ein-/ausschalten
CALO_INIT_API	Initialisierung der API-Umgebung
CALO_LOG_INIT_MEMORY	Protokollschnittstelle initialisieren
CALO_LOG_READ_MESSAGES	Meldungen des aktuellen Protokolleintrags lesen
CALO_MSG_APPEND_DB_LOG	Gesammelte Meldungen in das CAPI-Protokoll eintragen

CACL

Gruppe: Klassifikation

CACL_CLASSIFICATION_SAVE	Klassifizierung: Sicherungsbaustein
CACL_CLASS_ALLOCATION_MAINT	Klassifizierung: Zuordnungen von Klassen zu Klassen pflegen
CACL_CLASS_READ_ALLOCATIONS	Klassifizierung: Zuordnungen zur Klasse lesen
CACL_CLASS_READ_ALLOCATIONS_TD	Klassifizierung: Zuordnungen zur Klasse-TD lesen
CACL_CLASS_READ_VALIDATION	Klassifizierung: Bewertungen zu Klassen lesen
CACL_CLASS_VALIDATION_MAINT	Klassifizierung: Bewertung Klasse zu Klasse pflegen
CACL_OBJECT_ALLOCATION_MAINT	Klassifizierung: Zuordnungen zum Objekt pflegen
CACL_OBJECT_READ_ALLOCATIONS	Klassifizierung: Zuordnungen zum Objekt lesen
CACL_OBJECT_READ_VALIDATION	Klassifizierung: Bewertungen zum Objekt lesen
CACL_OBJECT_VALIDATION_MAINT	Klassifizierung: Bewertung zum Objekt pflegen



Die Funktionsgruppe CACL darf nicht im Bereich "Beziehungswissen" aufgerufen werden.

Liste der vorhandenen APIs

CARD**Gruppe: Daten der Variantenkonfiguration lesen**

CARD_CNET_CONSTRAINT_READ	Constraint eines Constraintnetzes lesen
CARD_CONSTRAINT_NET_READ	Constraintnetz lesen
CARD_CON_PROFILE_READ	Konfigurationsprofil lesen
CARD_DEPENDENCY_READ	Globales Beziehungswissen lesen
CARD_FUNCTION_READ	Variantenfunktion lesen
CARD_TABLE_READ_ENTRIES	Inhalt einer Variantentabelle lesen
CARD_TABLE_READ_STRUCTURE	Struktur einer Variantentabelle lesen
CARD_TABLE_SELECT_LINES	Variantentabellenzeilen mit bestimmter Merkmalausprägung selektieren

CACTR**Gruppe: Merkmaldaten lesen**

CARD_CHARACTERISTIC_READ	Attribute und zulässige Werte eines Merkmals lesen
CARD_CHAR_READ_ALLOC	Liste des Beziehungswissens zum Merkmal lesen
CARD_CHAR_READ_DEP	Beziehungswissen zum Merkmal lesen
CARD_CHAR_VAL_READ_ALLOC	Liste des Beziehungswissens zum Merkmalwert lesen
CARD_CHAR_VAL_READ_DEP	Beziehungswissen zum Wert eines Merkmals lesen

CACLR**Gruppe: Klassendaten lesen**

CARD_CLASS_CHAR_READ_ALLOC	Liste des Beziehungswissens zum überschriebenen Merkmal lesen
CARD_CLASS_CHAR_READ_DEP	Beziehungswissen eines überschriebenen Merkmals lesen
CARD_CLASS_CHAR_VAL_READ_ALLOC	Liste des Beziehungswissens zum überschriebenen Merkmalwert lesen
CARD_CLASS_CHAR_VAL_READ_DEP	Beziehungswissen eines überschriebenen Merkmalwerts lesen
CARD_CLASS_READ	Merkmale und Attribute einer Klasse lesen
CARD_CLASS_READ_ALLOC	Liste des Beziehungswissens zur Klasse lesen
CARD_CLASS_READ_ATTR	Attribute einer Klasse lesen (ohne Merkmale)

Liste der vorhandenen APIs

CARD_CLASS_READ_CHARACTS	Merkmale einer Klasse lesen (inkl. ererbten Merkmalen und Attributen)
CARD_CLASS_READ_DEP	Beziehungswissen zur Klasse lesen

CAMA

Gruppe: Datenpflege für die Variantenkonfiguration

CAMA_CNET_CONSTRAINT_MAINTAIN	Constraint innerhalb eines Netzes pflegen
CAMA_CONSTRAINT_NET_MAINTAIN	Constraintnetz pflegen
CAMA_CON_PROFILE_MAINTAIN	Konfigurationsprofil pflegen
CAMA_DEPENDENCY_MAINTAIN	Beziehungswissen einer globalen Beziehung pflegen
CAMA_FUNCTION_MAINTAIN	Variantenfunktion pflegen
CAMA_TABLE_MAINTAIN_ENTRIES	Inhalt einer Variantentabelle pflegen
CAMA_TABLE_MAINTAIN_ENTRI_LINE	Inhalt einer Variantentabellenzeile pflegen
CAMA_TABLE_MAINTAIN_STRUCTURE	Struktur einer Variantentabelle pflegen
CAMA_TABLE_MAINTAIN_VALUES	Inhalt einer Variantentabelle pflegen (ohne Beziehungswissen)
CAMA_TABLE_SAVE	Alle Daten einer Variantentabelle sichern

CACTM

Gruppe: Merkmaldaten pflegen

CAMA_CHARACTER_MAINTAIN	Merkmal pflegen (anlegen, ändern, löschen)
CAMA_CHARACTER_SAVE	Alle Daten des Merkmals sichern
CAMA_CHAR_ALLOCATE_GLOB_DEP	Globales Beziehungswissen an einem Merkmal pflegen
CAMA_CHAR_DEL_DEP	Beziehungswissen eines Merkmals löschen
CAMA_CHAR_MAINTAIN_DEP	Lokales Beziehungswissen an einem Merkmal pflegen
CAMA_CHAR_SINGL_MAINTAIN	Einzelnes Merkmal pflegen (anlegen, ändern, löschen)
CAMA_CHAR_VAL_ALLOCAT_GLOB_DEP	Globales Beziehungswissen an einem Merkmalwert pflegen
CAMA_CHAR_VAL_DEL_DEP	Beziehungswissen an einem Merkmalwert löschen
CAMA_CHAR_VAL_MAINTAIN_DEP	Lokales Beziehungswissen an einem Merkmalwert pflegen

Liste der vorhandenen APIs

CACLM**Gruppe: Klassendaten pflegen**

CAMA_CLASS_ALLOCAT_GLOB_DEP	Globales Beziehungswissen einer Klasse zuordnen
CAMA_CLASS_CHAR_ALLOC_GLOB_DEP	Globales Beziehungswissen dem Merkmal einer Klasse zuordnen
CAMA_CLASS_CHAR_DEL_DEP	Beziehungswissen eines überschriebenen Merkmals löschen
CAMA_CLASS_CHAR_MAINTAIN_DEP	Lokales Beziehungswissen zum Merkmal einer Klasse pflegen
CAMA_CLASS_CHVAL_ALL_GLOB_DEP	Globales Beziehungswissen dem Merkmalwert einer Klasse zuordnen
CAMA_CLASS_CHVAL_DEL_DEP	Beziehungswissen eines überschriebenen Merkmals löschen
CAMA_CLASS_CHVAL_MAINTAIN_DEP	Lokales Beziehungswissen zum Merkmalwert einer Klasse pflegen
CAMA_CLASS_DEL_DEP	Beziehungswissen einer Klasse löschen
CAMA_CLASS_MAINTAIN	Klasse pflegen (anlegen, ändern, löschen)
CAMA_CLASS_MAINTAIN_DEP	Lokales Beziehungswissen zur Klasse pflegen
CAMA_CLASS_SAVE	Alle Daten einer Klasse sichern
CAMA_CLASS_SNGL_MAINTAIN	Klasse pflegen (anlegen, ändern, löschen)
CAMA_WWS_CLASS_MAINTAIN	Mehrere Klassen gleichzeitig pflegen (anlegen, ändern, löschen)



Die folgenden APIs der Funktionsgruppen CAVC_... ersetzen die APIs der Gruppe CACO.

CAVC_OBJECTS**Gruppe: Initialisierung und Abschluß einer Konfiguration**

CAVC_O_ORDER_BOM_CANCEL	Bearbeitung der Auftragsstückliste abbrechen
CAVC_O_ORDER_BOM_INIT	Auftragsstückliste initialisieren
CAVC_O_ORDER_BOM_SAVE	Auftragsstückliste sichern

Liste der vorhandenen APIs

CAVC_CFG

Gruppe: Behandlung der kompletten Konfiguration

CAVC_C_CHECK	Konfiguration prüfen
CAVC_C_EXECUTE	Konfiguration durchführen
CAVC_C_GET_BOM_ITEM_DATA	Liefert zu allen Instanzen die Positionsdaten
CAVC_C_GET_INSTANCES	Alle Instanzen einer Konfiguration
CAVC_C_SET_LEVEL_BOM_EXPLOSION	Tiefe der Stücklistenauflösung festlegen
CAVC_C_SET_MODE_FIXING	Modus für Stücklistenfixierung festlegen
CAVC_C_SET_MODE_INSTANTIATION	Modus für Instanziierung festlegen

CAVC_INST

Gruppe: Behandlung von einzelnen Instanzen

CAVC_I_CHANGE_BOM_ITEM_DATA	Eine bestehende Instanz ändern (Stücklistenposition)
CAVC_I_CHARS_DEL_VALUES	Wertsetzungen löschen
CAVC_I_CHARS_DEL_VALUES_USER	Alle Wertsetzungen mit Benutzerrechtfertigung löschen
CAVC_I_CHARS_GET_ALLOWED_VALS	Erlaubte Werte zu Merkmalen lesen
CAVC_I_CHARS_GET_ASSIGNED_VALS	Aktuelle Wertsetzungen lesen
CAVC_I_CHARS_GET_ATTRIBUTES	Merkmalattribute lesen
CAVC_I_CHARS_GET_DEFAULT_VALS	Vorschlagswerte zu Merkmalen lesen
CAVC_I_CHARS_GET_VALID_VALS	Aktuell gültige Werte zu Merkmalen lesen
CAVC_I_CHARS_SET_VALUES	Wertsetzung zu Merkmalen
CAVC_I_CHECK	Instanz prüfen
CAVC_I_DELETE_BOM_ITEM	Instanz löschen (Stücklistenposition)
CAVC_I_DUPLICATE_BOM_ITEM	Instanz duplizieren (Stücklistenposition)
CAVC_I_FIX_BOM_ITEM	Stückliste entsprechend dem gesetzten Modus fixieren
CAVC_I_GET_BOM_HEADER_DATA	Kopfdaten einer Instanz lesen (Stücklistenkopf)
CAVC_I_GET_BOM_ITEM_DATA	Daten einer Instanz lesen (Stücklistenposition)
CAVC_I_GET_BOM_ITEM_DATA_LIST	Positionsdaten zu mehreren Instanzen lesen (Stücklistenpositionen)
CAVC_I_GET_CHARS	Merkmale einer Instanz lesen
CAVC_I_GET_CLASS_INFO	Informationen zum Klassenknoten lesen (Spezialisierung, Objektliste, Basisdaten)

Liste der vorhandenen APIs

CAVC_I_GET_PARENT	Zur angegebenen Instanz die Vaterinstanz liefern
CAVC_I_GET_PROFILE	Das verwendete Konfigurationsprofil einer Instanz ermitteln
CAVC_I_GET_STATUS	Status einer Instanz lesen
CAVC_I_INCONSISTENCIES	Inkonsistenzmeldungen einer Instanz lesen
CAVC_I_INSERT_BOM_ITEM	Eine neue Instanz hinzufügen (Stücklistenposition)
CAVC_I_INSTANCIATE_BOM_ITEM	Instanziierung entsprechend dem gesetzten Modus durchführen
CAVC_I_SELECT_CHILDS	Zu der angegebenen Instanz alle direkten Instanzen liefert
CAVC_I_SET_FIELDS_INST	Datenfelder zur Instanziierung füllen
CAVC_I_SET_STATUS	Status einer Instanz setzen

CAVC_DIALOG

Gruppe: Dialogfunktionen

CAVC_D_STRUCTURE_WINDOW	Konfigurationsstruktur bewerten (im Dialog)
-------------------------	---

CCAP

Gruppe: Änderungsstammdaten lesen und pflegen

CCAP_ALT_DATE_CHANGE	API Änderungsdienst: Datumselement ändern
CCAP_ALT_DATE_CREATE	API Änderungsdienst: Datumselement anlegen
CCAP_ALT_DATE_DELETE	API Änderungsdienst: Datumselement löschen
CCAP_ASSIGN_OBJECT_TO_ALT_DATE	API Änderungsdienst: Zuordnung Objekt zu Datumselement pflegen
CCAP_ECN_CREATE	API Änderungsdienst: Änderungsstamm anlegen (komplett)
CCAP_ECN_HEADER_CHANGE	API Änderungsdienst: Änderungsnummer Kopf und Objekttypdaten ändern
CCAP_ECN_HEADER_CREATE	API Änderungsdienst: Änderungsnummer Kopf und Objekttypdaten anlegen
CCAP_ECN_HEADER_DELETE	API Änderungsdienst: Änderungsnummer löschen

Liste der vorhandenen APIs

CCAP_ECN_HEADER_READ	API Änderungsdienst: Änderungsnummer auf Existenz prüfen
CCAP_ECN_HEADER_READ_MULTIPLE	API Änderungsdienst: Mehrere Änderungsnummern auf Existenz prüfen
CCAP_ECN_MAINTAIN	API Änderungsdienst: Änderungsstamm pflegen (komplett)
CCAP_REV_LEVEL_MAINTAIN	API Änderungsdienst: Revisionsstand erzeugen

CSAP

Gruppe: Stücklistendaten lesen und pflegen

CSAP_BOM_ITEM_MAINTAIN	API Stücklisten: Stücklistenposition bearbeiten
CSAP_DOC_BOM_CREATE	API Stücklisten: Dokumentstückliste anlegen
CSAP_DOC_BOM_MAINTAIN	API Stücklisten: Dokumentstückliste pflegen
CSAP_DOC_BOM_READ	API Stücklisten: Dokumentstückliste anzeigen
CSAP_MAT_BOM_ALLOC_CREATE	API Stücklisten: Materialstückliste anlegen
CSAP_MAT_BOM_CLOSE	API Stücklisten: Abschließen/Buchen Materialstückliste nach Änderung
CSAP_MAT_BOM_CREATE	API Stücklisten: Materialstückliste anlegen
CSAP_MAT_BOM_DELETE	API Stücklisten: Materialstückliste löschen
CSAP_MAT_BOM_ITEM_SELECT	API Stücklisten: Positionen einer Materialstückliste selektieren
CSAP_MAT_BOM_MAINTAIN	API Stücklisten: Materialstückliste pflegen
CSAP_MAT_BOM_OPEN	API Stücklisten: Eröffnen Materialstückliste für Änderungen
CSAP_MAT_BOM_READ	API Stücklisten: Materialstückliste anzeigen
CSAP_MAT_BOM_SELECT	API Stücklisten: Materialstückliste(n) selektieren

Allgemeine Abarbeitungsregeln

Allgemeine Abarbeitungsregeln

Für alle Objekte wie z. B. Merkmal, Klasse, Beziehungswissen gibt es Pflegebausteine. Mit diesen APIs kann das entsprechende Objekt angelegt, geändert und auch gelöscht werden.

Um Objekte über APIs pflegen zu können, müssen an die Schnittstelle bestimmte Daten als Einzelparameter oder in internen Tabellen übergeben werden.

Beispiel: Merkmal anlegen

Sie möchten z. B. die Merkmale **Farbe** und **Länge** anlegen, mit der deutschen Bezeichnung 'O-Farbe' bzw. 'Stangenlänge' und dem Merkmalstatus '2'. Dazu muß die interne Tabelle *Merkmalattribute* folgende Informationen enthalten:

Merkmalname	=	'FARBE'
Status	=	'2'
Datentyp	=	'CHAR'
Anzahl Stellen	=	'9'

Merkmalname	=	'LAENGE'
Status	=	'2'
Datentyp	=	'NUM'
Anzahl Stellen	=	'4'
Negative Werte	=	'X'

Außerdem muß die Tabelle *Bezeichnungen* folgende Informationen enthalten:

Merkmalname	=	'FARBE'
Sprache	=	'D'
Bezeichnung	=	'O-Farbe'

Merkmalname	=	'LAENGE'
Sprache	=	'D'
Bezeichnung	=	'Stangenlänge'

Mit diesen Informationen kann der Funktionsbaustein CAMA_CHARACTER_MAINTAIN aufgerufen und ausgeführt werden. Das Merkmal wird dann im R/3-System angelegt.

Beispiel: Merkmal ändern

Das vorhandene Merkmal **Farbe** soll geändert werden. Sie möchten den Merkmalstatus von 2 auf 1 setzen und die deutsche Merkmalbezeichnung von 'O-Farbe' in 'Oberfl.-Farbe' ändern.

Allgemeine Abarbeitungsregeln

Dazu rufen Sie den Funktionsbaustein CAMA_CHARACTER_MAINTAIN auf mit folgendem Input für die Tabelle *Merkmalattribute*:

Merkmalname	=	'FARBE'
Status	=	'1'

Die interne Tabelle *Bezeichnungen* bekommt folgende Einträge:

Merkmalname	=	'FARBE'
Sprache	=	'D'
Bezeichnung	=	'O-Farbe'
Löschkennzeichen	=	'X'

Merkmalname	=	'FARBE'
Sprache	=	'D'
Bezeichnung	=	'Oberfl.-Farbe'
Löschkennzeichen	=	SPACE

Zuerst wird die alte Bezeichnung gelöscht, dann wird der neue Wert eingefügt.



1. Falls Sie ein Objekt ändern, das es noch nicht gibt, wird das Objekt angelegt.
2. Falls Sie ein Objekt bzw. ein Element löschen, das es noch nicht gibt, wird diese Angabe ignoriert.

Systemmeldungen

Systemmeldungen

Exceptions/Ausnahmen

Exceptions werden für alle Fehler benutzt und grundsätzlich am Ende des APIs gesetzt, damit unterschiedliche Fehler aufgedeckt und protokolliert werden können.

Das API überprüft alle Eingaben. Wenn Fehler auftreten, werden diese protokolliert und die Ausnahme ERROR gesetzt.



In Ausnahmefällen kann eine Exception auch gesetzt werden, ohne daß alle Prüfungen durchgeführt werden.

In der Regel wird im Fehlerfall nur die Ausnahme ERROR gesetzt, und zwar dann, wenn die Bearbeitung z.B. aufgrund fehlender Daten abgebrochen werden musste.

Einige Bausteine verwenden auch die Ausnahme oder ein Returncode WARNING, falls eine Warnmeldung in das Protokoll geschrieben wurde.

Zusätzlich zur Exception wird für jeden entdeckten Fehler ein Eintrag in das Protokoll durchgeführt.

Protokoll

Die Meldungen, wie sie im Dialog erscheinen würden, werden in ein Datenbank-Protokoll eingetragen. Es werden alle Arten von Meldungen (Fehler, Warnungen und auch Erfolgsmeldungen) eingetragen. Dadurch werden alle Abläufe und auch komplexe Fehlerfälle erläutert.

API zum Lesen des Meldungsprotokolls

Das erzeugte Protokoll kann über einen eigenen API-Baustein gelesen werden, bevor es eventuell auf die Datenbank geschrieben wird. Die Meldungen und einige Meldungsattribute werden in einer internen Tabelle übergeben. Nach jedem von Extern aufgerufenen API stehen die gesammelten Meldungen (in abgemischter Form) diesem API zur Verfügung. Der API-Anwender kann programmintern die Meldungsidentifikation und den Meldungstyp auswerten und die Meldungen in seiner Benutzerschnittstelle präsentieren.

Siehe auch [Allgemeine APIs \[Seite 33\]](#).

Nomenklatur für die APIs

Alle verwendeten Namen sollten sprechend und auf englisch sein. Dies gilt insbesondere auch für Feld- und Strukturnamen. In der Schnittstelle werden nur CHAR und NUMC Felder benutzt.

Bei der Vergabe der Namen für APIs wird nach folgendem Schema vorgegangen:

xxxx_Objekt_Aktion_Ergebnis

Der Name besteht aus folgenden Einzelementen:

•	xxxx	Funktionsgruppe
•	Objekt	Objekt/ Schlüssel (Was habe ich?)
		(vgl. Thema API Objekte [Seite 25])
•	Aktion	Aktion, die ausgeführt werden soll, z.B. Read
		(vgl. Thema API Aktionen [Seite 26])
•	Ergebnis	das erwartete Ergebnis, z.B. Merkmale einer Klasse



CAMA_CLASS_MAINTAIN
CSAP_MAT_BOM_CREATE

Identifikation der einzelnen Objekte

Identifikation der einzelnen Objekte

Die Identifikation der Objekte, d.h. der Objektschlüssel, ist im Rahmen der Klassifizierung sehr relevant. Dort muß dem Baustein über den Objekttyp und den Objektschlüssel mitgegeben werden, welche Objekte klassifiziert werden sollen. Der Importparameter OBJECT_TYPE gibt die Objekttable an. Als Objekttable tragen Sie z.B. MARA für Materialien ein.

Im Customizing des *Klassensystems* können Sie die Keyfelder, über die die einzelnen Objekte identifiziert werden, für den Importparameter OBJECT_TYPE ableiten. Wählen Sie den Bereich *Klassen* und die IMG-Aktion *Objektschlüssel pflegen*. Über einen Doppelklick auf den gewünschten Tabellennamen erfahren Sie die Keyfelder für die Tabelle.

Für ein Dokument könnte die Objekttable z.B. folgendermaßen aussehen:

<u>Field</u>	<u>Value</u>
DOKAR	'DRAW'
DOKNR	'123'
DOKVR	'2'
DOKTL	'1'

API Objekte

Folgende Namen können für das angesprochene Objekt im Namen für einen Funktionsbaustein verwendet werden:

BOM	Bill of Material, Stückliste
CH_VALUE, VALUE	Merkmalwert
CHARACTERISTIC	Merkmal
CLASS	Klasse
CONFIGURATOR	der Variantenkonfigurator
CON_RESULT	Konfigurationsergebnis
DB_LOG	Meldungsprotokoll in der Datenbank
ECN_HEADER	Änderungsstamm Kopfdaten
MAT_BOM	Zuordnung Material - Stückliste
MATERIAL	Materialnummer
MSG	Meldung vom API an das aufrufende Programm
OBJ_TYPE	Objekttyp
OBJECT	R/3-Objekt, z.B. Material
SEL_RESULT	Suchergebnis, nach einer Suche im Klassensystem
INSTANCE	Instanz als Identifikator eines Objektes einer
	Konfigurationsstruktur
(VARIANT-) TABLE	Variantentabelle

API Aktionen

API Aktionen

Folgende Namen können für die angesprochenen Aktionen im Funktionsbausteinamen verwendet werden:

- **Create**
Anlegen eines neuen (API-)Objektes.
- **Insert**
Einfügen, welches auch ein Anlegen ist, bei der die Position in einer Liste wichtig ist.
Bsp. Merkmalwert in eine vorhandene Werte-Liste einfügen.
- **Append**
In einer Liste hinten anfügen.
- **Read**
Objektdateien lesen. Bsp. Klassenattribute lesen.
- **Check**
Prüfen
- **Change**
Ändern wird in der Regel durch ein Löschen und ein Schreiben durchgeführt. Das Schreiben erfolgt dann mit dem Maintain-Baustein.
- **Init**
Initialisieren
- **Prepare**
Vorbereiten
- **Fix**
Fixieren
- **Define**
Definieren
- **Execute**
Ausführen
- **Activate**
Aktivieren
- **Deactivate**
Deaktivieren
- **Convert**
Konvertieren
- **Delete**

- Löschen
- **Select**
Selektieren bzw. Suchen

Zuordnung von Beziehungswissen

Zuordnung von Beziehungswissen

Verwendung

Wie im Dialog können Sie auch über APIs folgenden Objekten Beziehungswissen zuordnen:

- Merkmal
- Merkmalwert
- Stücklistenposition
- Konfigurationsprofil

Es gibt 6 Arten von Beziehungswissen:

1	CON	Constraint
2	PRE	Vorbedingung
3	ACT	Aktion
4	SEL	Auswahlbedingung
5	CNET	Constraintnetz
6	PROC	Prozedur

Die Angaben der Beziehungsarten sind kompatibel, d.h. Sie können sowohl die Nummer als auch das Kürzel angeben. Die Ausgabe der Lesebausteine erfolgt stets mit den externen Kürzeln.

Die Pflege und das Lesen des Beziehungswissens erfolgt nach 2 verschiedenen Methoden für folgende Funktionsgruppen:

- CSAP
- CARD und CAMA

Regeln für die Bausteine der Gruppe CSAP



Auch den Bausteinen CAMA_CON_PROFILE_MAINTAIN (Konfigurationsprofil pflegen) und CAMA_TABLE_MAINTAIN_ENTRIES (Tabelleninhalt pflegen) wird nach diesem Schema Beziehungswissen zugeordnet.

Für die einzelnen Objekte gibt es Pflegebausteine, die in der Schnittstelle fünf Tabellen für die Daten des Beziehungswissens enthalten. Über diese Tabellen können Sie lokales und globales Beziehungswissen zuordnen:

- Da eine lokale Beziehung erst beim Sichern einen internen Namen bekommt, geben Sie beim Anlegen einen externen Namen an, über den die abhängigen Daten, die zu einer Beziehung gehören, in den verschiedenen Tabellen angesprochen werden.
- Wenn Sie globales Beziehungswissen zuordnen möchten, geben Sie den Namen des Beziehungswissens im Feld *Interner Beziehungsname* an. Die Zuordnung von globalem Beziehungswissen nehmen Sie über die Tabelle DEP_ORDER vor.

Zuordnung von Beziehungswissen

In diesen Tabellen werden folgende Daten abgelegt:

- Tabelle *Beziehungswissen Basisdaten* (DEP_DATA)

In dieser Tabelle werden die Basisdaten einer Beziehung übergeben.

 - Interner Beziehungsname
 - Externer Beziehungsname
 - Beziehungsart
 - Status
 - Beziehungsgruppe
 - Berechtigungsprofil
 - Löschkennzeichen

- Tabelle *Beziehungswissen Bezeichnung* (DEP_DESCR)

In dieser Tabelle werden die sprachabhängigen Bezeichnungen der Beziehung übergeben.

 - Interner Beziehungsname
 - Externer Beziehungsname
 - Sprache
 - Bezeichnung
 - Löschkennzeichen

- Tabelle *Zugeordnetes Beziehungswissen* (DEP_ORDER)
 - Interner Beziehungsname
 - Externer Beziehungsname
 - Zähler für Sortierreihenfolge der Objekte

In der Tabelle *Zugeordnetes Beziehungswissen* wird Ihnen das Beziehungswissen angezeigt, das einem Objekt zugeordnet wurde. Wenn Sie eine Zuordnung löschen möchten, können Sie in dieser Tabelle oder der Tabelle *Basisdaten* das Löschkennzeichen setzen. Bei globalen Beziehungswissen wird nur die Zuordnung gelöscht, bei lokalem Beziehungswissen wird die ganze Beziehung gelöscht. Außerdem können Sie in dieser Tabelle über das Feld DEP_LINENO die Sortierreihenfolge für Prozeduren angeben.

- Tabelle *Beziehungswissen Quelltext* (DEP_SOURCE)

In dieser Tabelle werden die Quelltexte einer Beziehung übergeben.

 - Interner Beziehungsname
 - Externer Beziehungsname
 - Zeilennummer
 - Zeile für Quelltext

- Tabelle *Beziehungswissen Dokument* (DEP_DOC)

Zuordnung von Beziehungswissen

- Interner Beziehungsname
- Externer Beziehungsname
- Sprache
- Zeilennummer
- Textformat

Zu Beziehungswissen können Sie Langtexte erfassen. Im Feld TXT_FOR der Tabelle *Dokument* können Sie Angaben im SAPscript Format machen:

- AS Standardabsatz
- B1 Liste erster Stufe
- N1 numerierte Liste etc.

Regeln für die Bausteine der Gruppen CARD und CAMA

Um lokales Beziehungswissen zuzuordnen, müssen die Angaben vollständig sein, da das lokale Beziehungswissen im Gegensatz zu globalem Beziehungswissen keinen für den Anwendungsprogrammierer sichtbaren Namen erhält.

Jeder weitere Aufruf erzeugt einen neuen Wissensbaustein. Wird das Löschkennzeichen gesetzt, so werden alle lokalen Wissensbausteine gelöscht. Als lokales Beziehungswissen können Sie Aktionen, Prozeduren, Auswahlbedingungen und Vorbedingungen anlegen. Constraints können nicht lokal definiert werden.

Beim Anlegen von lokalem Beziehungswissen müssen folgende Angaben gemacht werden:

- Sprache
- Bezeichnung
- Status
- Art des Beziehungswissen
- mindestens eine Zeile Coding

Fehlerursachen stehen im Protokoll.

Die Zuordnung von globalen Beziehungswissen erfolgt über den Namen, der beim Anlegen des Beziehungswissen vergeben wird. Dieser Name bleibt während des Gültigkeitszeitraums des Beziehungswissens erhalten und ist eindeutig. Die Zuordnung erfolgt durch Angabe des Namens in einer internen Tabelle. Zuordnungen können einzeln erzeugt oder gelöscht werden.

Beim Lesen von Beziehungswissen wird das gesamte lokale Wissen mit allen verfügbaren Daten (Bezeichnung, Source usw.) bereitgestellt. Jeder Aufruf liefert einen Baustein des Beziehungswissens. Das globale Wissen erscheint als Liste von Zuordnungen in einer gesonderten Tabelle.

Der Aufbau der einzelnen Tabellen entspricht im Prinzip dem der Methode der CSAP-Bausteine. Es entfallen jedoch alle Namensfelder, und die Basisdaten des Beziehungswissens werden als einzelne Struktur und nicht als Tabelle übergeben.

Generierung von ABAP Coding

Über die Transaktion GENC können Sie sich aus den API Funktionsbausteinen von SAP eigenes ABAP Coding generieren. Gehen Sie dabei folgendermaßen vor:

1. Geben Sie auf dem Einstiegsbild den Namen Ihres Programms an.
2. Auf dem folgenden Bild wird Ihnen die Funktionsstruktur Ihres Programms angezeigt.

In der Struktur ist immer der Funktionsbaustein CALO_INIT_API enthalten (siehe [Initialisierung der APIs \[Seite 34\]](#)). Er kann nicht gelöscht werden. In der Struktur wird für diesen Funktionsbaustein die Bezeichnung *Initialize Logfile* angegeben.

Der Anfang und das Ende eines Funktionsbausteins wird über bestimmte Kürzel deutlich gemacht. Die Bezeichnung für einen Funktionsbaustein, die in der Struktur angegeben werden soll, wird nach dem Kürzel angegeben, das den Anfang eines Funktionsbausteins bezeichnet, und kann gegebenenfalls geändert werden:

***&BPAPILogfile&:Initialize Logfile**

perform api_init_logfile.

*&EPAPILogfile&

Dem Programm können Sie weitere, von SAP bereitgestellte API Bausteine über die Funktion *Bearbeiten* → *Neue Funktionen* zuordnen. Sie erhalten ein Dialogfenster, auf dem die APIs nach den Bereichen, zu denen sie gehören, gegliedert sind. Mit Doppelklick können Sie einen Funktionsbaustein übernehmen.

Bei Funktionsbausteinen mit optionalen Daten können Sie angeben, welche Parameter an der Schnittstelle übergeben werden sollen.

Beim Pflegen eines Merkmals können Sie z. B. festlegen, ob die Basisdaten, die Merkmalbezeichnung, Werte oder Wertebezeichnungen als Parameter an den Schnittstellen übergeben werden sollen.

Geben Sie außerdem an, ob über den Baustein ein Merkmal angelegt oder geändert werden soll und generieren Sie dann den Funktionsbaustein, um ihn in Ihr Programm zu übernehmen.

3. Über die Funktion *Programm* können Sie den Quelltext anzeigen oder ändern. Sie können z. B. eigene Funktionen in das Programm aufnehmen. Wenn diese Funktionen auch in der Funktionsstruktur angegeben werden sollen, muß Anfang und Ende der Funktion über folgende Kürzel gekennzeichnet sein:

*&BPUSR&-	Anfang der Funktion
	Hinter diesem Kürzel geben Sie den Namen an, der in der Funktionsstruktur erscheint
*&EPUSR&-	Ende der Funktion

4. Die Daten, die das Programm zur Verarbeitung benötigt, werden über Platzhalter angegeben (<Platzhalter>). Die Platzhalter müssen durch konkrete Werte ersetzt werden.

Falls z. B. bei dem Baustein *Merkmal pflegen* als Parameter für die Schnittstelle die Basisdaten und die Merkmalwerte angegeben wurden, müssen folgende Platzhalter ersetzt werden:

Generierung von ABAP Coding

```
* Define basic data
perform ct_basics tables ct_characts
                        using      '<ct_name>'
                                '<ct_type>'
                                '<ct_length>'
                                '<ct_status>' .

* Define value
perform ct_value tables ct_char_vals
                        using      '<ct_name>'
                                '<ct_value>'
                                '<ct_default>' .
```

5. Über die Funktion *Programm* → *Log-Einstellungen* legen Sie fest, ob für jeden Baustein die Protokolleinträge auf der Datenbank geschrieben werden oder ob alle Meldungen gesammelt und erst am Ende gelesen werden.

Bei der Einstellung *Meldungen sammeln* muß der Funktionsbaustein CALO_MSG_APPEND_DB_LOG aufgerufen werden, um die Protokolleinträge auf die Datenbank zu schreiben (siehe [APIs für die Protokollsteuerung \[Seite 36\]](#)).

Allgemeine APIs (CALO)

Allgemeine APIs beziehen sich nicht auf die Daten einer bestimmten Anwendung sondern auf allgemeine Funktionen. Über diese APIs wird die API-Umgebung initialisiert, die Protokollsteuerung festgelegt und auf das Protokoll zugegriffen.

Initialisierung der APIs

Initialisierung der APIs

Für die Initialisierung der API-Umgebung steht Ihnen der Funktionsbaustein **CALO_INIT_API** zur Verfügung. Alle Übergabeparameter sind mit Defaultwerten belegt.

Wenn APIs aus den Bereichen Klassen, Varianten, Stücklisten und Änderungsdienst benutzt werden sollen, **muß** dieser Baustein als erster aufgerufen werden.

INPUT:	FLAG_DB_LOG_ON (Datenbankprotokoll eingeschaltet)	Default: X
	FLAG_MSG_ON (Übergabe der Meldungen in interne Tabelle)	Default: X
	FLAG_API_API_CALL_ON (API wird von API aufgerufen → Meldungen sammeln)	Default: Space
	FLAG_COLLECT_MSG_ON Meldungen sammeln	Default: Space
	EXTERNAL_LOG_NO Externe Log-Nr (siehe Transakt. SLG1)	Default: API
	DEL_LOG_AFTER_DAYS DB Log wird nach n-Tagen gelöscht	Default: 10
	DATA_RESET_SIGN Rücknahmekennzeichen (Setzt Datenfeld auf Initialwert zurück)	Default: Space

Wenn eines der beiden Kennzeichen *Datenbankprotokoll* oder *Übergabe der Meldungen* eingeschaltet ist, werden Meldungen, die von den APIs ausgegeben werden, zunächst in ein lokales Gedächtnis geschrieben.

Falls das Kennzeichen *Datenbankprotokoll* markiert ist und das Kennzeichen *Meldungen_sammeln* nicht eingeschaltet ist, schreibt jeder API einen Protokolleintrag in die Datenbank, in dem alle Meldungen enthalten sind.

Wenn das Kennzeichen *Meldungen_sammeln* markiert ist, werden die Meldungen solange gesammelt, bis das Anwendungsprogramm das API "Protokoll Schreiben" aufruft.

Am Anfang von jedem API wird das lokale Protokoll-Gedächtnis initialisiert, wenn das Kennzeichen *Meldungen_sammeln* nicht gesetzt (= SPACE) ist.

Wenn dieses Kennzeichen markiert ist, muß der API-Anwender die Protokolleinträge bei Bedarf lesen und die Protokollschnittstelle initialisieren.

Wenn Sie die Anzahl der Tage eingegeben haben, nach denen das Datenbankprotokoll gelöscht werden kann, so bedeutet dies auch, daß das Protokoll nicht vor Ablauf dieser Frist gelöscht werden kann. Wenn Sie den Wert '-1' eingeben, ist es sofort löschar.

APIs für die Protokollsteuerung

APIs für die Protokollsteuerung**Protokollverfahren ein-/ausschalten: CALO_DB_LOG_ON_OFF**

Mit diesem Baustein kann das Protokollverfahren aus- und wieder eingeschaltet werden. In der Standardeinstellung wird das Protokollverfahren über den Baustein CALO_INIT_API eingeschaltet (siehe [Initialisierung der APIs \[Seite 34\]](#)).

- Wenn als Parameter für das Kennzeichen FLAG_DB_LOG_ON kein Wert (SPACE) gesetzt wird, kann das Protokollverfahren ausgeschaltet werden. Solange das Protokollverfahren ausgeschaltet ist, werden keine Meldungen in das Datenbankprotokoll eingetragen.
- Wird das Kennzeichen FLAG_DB_LOG_ON mit X markiert, wird das Protokollverfahren wieder eingeschaltet. Ab diesem Zeitpunkt werden die Meldungen intern gesammelt.



Falls das Protokoll-Objekt bzw. Subobjekt nicht gefunden werden kann, muß es mit Transaktion SLG0 angelegt werden:

Objekt:	CAPI
Subobjekt:	CAPI_LOG

INPUT:

FLAG_DB_LOG_ON	Default: X	(Datenbank-Protokoll einschalten)
----------------	------------	-----------------------------------

**Protokolleintrag in die Datenbank schreiben:
CALO_MSG_APPEND_DB_LOG**

Über diesen Baustein können Sie die im lokalen Gedächtnis gesammelte Meldungen in das Datenbank-Protokoll schreiben.

Anschließend wird das interne Gedächtnis initialisiert.



In Ihrem Programm werden viele APIs aufgerufen, und es sollte nicht jedes Mal ein Protokolleintrag in die Datenbank geschrieben werden. Die APIs sollten die Meldungen nur in das lokale Gedächtnis schreiben, und die gesammelten Meldungen, zu von Ihnen festgelegten Zeitpunkten, sollten Sie selbst in die Datenbank schreiben.

Vorgehensweise:

1. Der Baustein CALO_INIT_API wird aufgerufen und das Kennzeichen *Meldungen_sammeln* markiert.
2. Beliebige APIs werden aufgerufen, die alle Ihre Meldungen nur in das lokale Gedächtnis schreiben.

APIs für die Protokollsteuerung

- Der Baustein CALO_MSG_APPEND_DB_LOG wird dann aufgerufen, wenn ein Protokolleintrag (mit allen Meldungen) in die Datenbank geschrieben und das lokale Gedächtnis initialisiert werden soll.

Meldungen des aktuellen Protokolleintrags lesen: CALO_LOG_READ_MESSAGES

Mit diesem Baustein kann der Inhalt des lokalen Protokoll-Gedächtnisses gelesen werden.

Das lokale Protokoll-Gedächtnis wird hiermit nicht gelöscht, sondern erst,

- wenn das Protokoll auf die Datenbank geschrieben wird oder
- wenn Sie das API zum Initialisieren der Protokollschnittstelle aufrufen.

INPUT:	Klasse der Meldungen	Default: 4
	zulässige Werte:	
	1: Es werden nur besonders wichtige Protokolle gelesen.	
	2: Es werden nur wichtige Protokolle gelesen.	
	3: Es werden auch weniger wichtige Protokolle gelesen.	
	4: Es werden alle Protokolle gelesen.	
	Sprache	

Protokollschnittstelle initialisieren: CALO_LOG_INIT_MEMORY

Das lokale Protokoll-Gedächtnis wird hiermit gelöscht.

Das Protokoll

Das Protokoll kann mit der Transaktion SLG1 für Objekt CAPI, Unterobjekt CAPI_LOG ausgewertet werden. In einem Protokolleintrag sind alle Meldungen eines "Protokoll Schreiben"-Aufrufs enthalten.