



PREDICT

External Objects in Predict

Version 4.3.1

This document applies to Predict Version 4.3.1 and to all subsequent releases. Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

© June 2003, Software AG
All rights reserved.

The name Software AG and/or all Software AG product names are either trademarks or registered trademarks of Software AG. Other company and product names mentioned herein may be trademarks of their respective owners.

Table of Contents

External Objects in Predict - Overview	1
External Objects in Predict - Overview	1
Handling of External and Documentation Objects	2
Handling of External and Documentation Objects	2
What is an External Object	2
External Objects Owned by Predict	3
External Objects Not Owned by Predict	3
Connection of External and Documentation Objects	4
Disconnecting Objects	4
Impact of Connecting External and Documentation Objects	4
Overview of External Objects	5
Generation of External Objects	7
Generation of External Objects	7
Generation	8
Generation	8
Concepts of Generation	8
Connecting External Objects and Documentation Objects	9
Overview of Generation Functions	9
Calling Generation Functions	11
Calling Generation Functions with Commands	11
Generation Condition Codes	11
Generation Defaults	11
Preprocessor Force	12
Storage of External Objects Owned by Predict	13
Output / Storage Options for External Objects	13
Parameters Specifying the Form of Output	14
Storing External Objects with Entire System Server	16
Common Parameters	20
Adabas Version	21
Generate For Adabas	22
Generate For Adabas	22
ADACMP, ADAWAN, ADAFDU and ADALOD Definitions	23
ADACMP, ADAWAN, ADAFDU and ADALOD Definitions	23
Calling the Function	23
Additional ADALOD Parameters	27
Additional ADAFDU Parameters	28
Additional Parameters For UES Support	28
Generate ADACMP, ADAWAN, ADAFDU or ADALOD Definitions in Batch Mode	29
Sample Output	31
Adabas File	35
Adabas File	35
Calling the Function	35
Generate Adabas File in Batch Mode	38
Sample Output	40
Sample Output - Continued	41
ADAINV Definitions	42
ADAINV Definitions	42
Calling the Function	43
Generate ADAINV Definitions in Batch Mode	44
Sample Output	46
ADASCR Definitions	47
ADASCR Definitions	47
Calling the Function	47

Generate ADASCR Definitions in Batch Mode	48
Sample Output	49
Adabas Vista Translation Table	51
Adabas Vista Translation Table	51
Calling the Function	51
Generate Vista Table in Batch Mode	53
Adabas VSAM Bridge Transparency Table	54
Adabas VSAM Bridge Transparency Table	54
Prerequisites	54
Rules Applying to the Design of the Adabas Files	55
Calling the Function	55
Generate Transparency Table in Batch Mode	56
Examples	57
Sample File Definition	57
Sample Output	61
Adabas Table/Cluster/View	72
Adabas Table/Cluster/View	72
Calling the Function	73
Recommendations when Using Adabas Vista	75
Generate Adabas Table/View in Batch Mode	77
Generate for Natural	78
Generate for Natural	78
IMS User-Defined Fields	78
Rules Applying to the Generation of IMS UDFs	78
Calling the Function	79
Replace Verification Rule	81
Calling the Function	81
Data Definition Module	83
Rules Applying to the Generation of DDMs	83
Calling the Function	84
Generating DDMs for Use with IMS	87
Generating DDMs for Use with SQL	87
Generating a DDM for Use with Adabas Vista	88
Recommendations when Using Adabas Vista	88
Generate DDM in Batch Mode	89
Sample Output	92
Generate for DB2	94
Generate for DB2	94
Common Parameters for all DB2 Types	94
DB2 Database	94
Calling the Function	95
Generate DB2 Database in Batch Mode	96
Sample Output	96
DB2 Procedure/Function	97
Calling the Function	97
Generate DB2 Procedures/Functions in Batch Mode	100
DB2 Table/View	100
Calling the Function	101
Generate DB2 Table/View in Batch Mode	103
Sample File Definition	103
Sample Output	104
DB2 Storagegroup	105
Calling the Function	106
Generate DB2 Storagegroup in Batch Mode	107
Sample Output	107
DB2 Tablespace	108

Calling the Function	108
Generate DB2 Tablespace in Batch Mode	109
Sample Output	110
Generating for SQL	111
Generating for SQL	111
SQL CREATE Statements	111
Calling the Function	111
Generating SQL CREATE Statements in Batch Mode	113
Sample Output	114
Generating for OS/400	116
Generating for OS/400	116
Principles	116
Components	116
Mapping Rules	117
Defining an OS/400 File Object	120
Calling the Generate Function	123
Field Check Messages	128
Changing an OS/400 Database File	129
Generate For Third Generation Languages - 3GL	130
Generate For Third Generation Languages - 3GL	130
Additional Usage Information for Fields in a File	130
Assembler Copy Code	131
Assembler Copy Code	131
Calling the Function	131
Names in Assembler Copy Code	135
Assembler Names for Counter Fields - MC or PC	136
Assembler Names for Logical Fields - L	136
Assembler Names for Fields of Type AV	136
Assembler Names for Additionally Generated Indicator Fields	136
Field Format and Assembler Copy Code	136
Sample Output	138
C Include Code	140
C Include Code	140
Calling the Function	140
Generate C Include Code in Batch Mode	144
Names in C Include Code	146
C Names for Redefinition Fields - RE	146
C Names for Counter Fields (MC or PC)	146
C Names for Additionally Generated Indicator Fields	146
Field Format and C Include Code	147
Sample Output	151
COBOL Copy Code	154
COBOL Copy Code	154
Calling the Function	154
Generate COBOL Copy Code in Batch Mode	159
Names in COBOL Copy Code	160
COBOL Names for Counter Fields - MC or PC	161
COBOL Names for Multiple Fields - MU - Contained in Periodic Groups - PE	161
COBOL Names for Redefined Fields - RE in Periodic Groups - PE	161
COBOL Names for Logical Fields - L	161
COBOL Names for Record Buffer and Format Buffer	161
COBOL Names for Default Indexed by Name	161
COBOL Names for Additionally Generated Indicator Fields	161
Examples for the Generation of COBOL Field Names	162
Field Format and COBOL Copy Code	162
Automatically Generated Counter Fields	166

COBOL Copy Code for DB2 Tables/Views	167
Sample Output	167
FORTRAN Copy Code	169
FORTRAN Copy Code	169
Calling the Function	169
Generate FORTRAN Copy Code in Batch Mode	172
Names in FORTRAN Copy Code	174
FORTRAN Names for Multiple Fields - MU - Contained in Periodic Groups - PE	174
FORTRAN Names for Additionally Generated Indicator Fields	174
Examples for the Generation of FORTRAN Field Names	174
Field Format and FORTRAN Copy Code	175
Sample Output	176
PL/I Include Code	180
PL/I Include Code	180
Calling the Function	180
Generate PL/I Include Code in Batch Mode	185
Names in PL/I Include Code	186
PL/I Names for Counter Fields - MC or PC	186
PL/I Names for Additionally Generated Indicator Fields	186
Field Format and PL/I Include Code	187
Additional Notes	189
Sample Output	190
Sample File Definition	192
Sample File Definition	192
File Implementation	197
File Implementation	197
How this section is Organized	197
Concepts of File Implementation	197
Benefits of Implementation Plans	197
Overview of Activities	198
Executing Implementation Plans under Adabas Vista	198
Generating a DDM under Natural Security	199
Calling File Implementation Functions	199
The File Implementation Menu	199
Executing and Displaying an Implementation Plan in Batch Mode	200
File Implementation Functions	200
Add Implementation Plan - Code A	200
Copy Implementation Plan - Code C	202
Modify Implementation Plan - Code M	202
Rename Implementation Plan - Code N	205
Purge Implementation Plan - Code P	205
Display Implementation Plan - Code D	205
Extend Implementation Plan - Code E	206
Select Implementation Plan from List - Code S	206
Execute Implementation Plan - Code X	207
Reexecute Implementation Plan - Code R	208
Error Handling - Online	209
Incorporation	210
Incorporation	210
Concepts of Incorporation	211
Concepts of Incorporation	211
Overview of Incorporation Options	211
Overview of Incorporation Subfunctions	211
Connecting External and Predict Objects	212
Calling Incorporation Functions	213
Incorporation Functions are Executed in Two Steps	213

Step 1: Selecting the Objects to be Processed	213
Step 2: Processing Objects with Incorporation Functions	214
Overview of Function Codes and Commands	215
Using Incorporation Functions in Batch Mode	216
Incorporating Adabas Databases	217
Incorporating Adabas Databases	217
Prerequisites	217
Restrictions	217
Selecting Adabas Databases	217
Incorporation Functions for Adabas Databases	218
Compare - Code T	218
Connect - Code C	219
Incorporate - Code I	219
Incorporate Adabas Database in Batch Mode	219
Incorporating Adabas Files	221
Incorporating Adabas Files	221
Prerequisites and Restrictions	221
Selecting Adabas Files	221
Incorporation Functions for Adabas Files	223
Compare - Code T	223
Connect - Code C	223
Incorporate - Code I	223
Incorporating Adabas Files in Batch Mode	224
Incorporating DB2 Storagegroups	226
Incorporating DB2 Storagegroups	226
Prerequisites	226
Selecting DB2 Storagegroups	226
Incorporation Functions for DB2 Storagegroups	227
Compare - Code T	227
Connect - Code C	227
Incorporate - Code I	227
Incorporate DB2 Storagegroups in Batch Mode	227
Incorporating DB2 Databases	229
Incorporating DB2 Databases	229
Prerequisites	229
Selecting DB2 Databases	229
Incorporation Functions for DB2 Databases	230
Compare - Code T	230
Connect - Code C	230
Incorporate - Code I	230
Incorporate DB2 Databases in Batch Mode	231
Incorporating DB2 Tablespaces and SQL/DS DBspaces	232
Incorporating DB2 Tablespaces and SQL/DS DBspaces	232
Prerequisites	232
Selecting Tablespaces	232
Incorporation Functions for Tablespaces	233
Compare - Code T	233
Connect - Code C	233
Display Masters - Code M	233
Incorporate - Code I	234
Incorporate DB2 Tablespaces in Batch Mode	234
Incorporating DB2 and SQL/DS Tables/Views	235
Incorporating DB2 and SQL/DS Tables/Views	235
Prerequisites	235
Selecting DB2 and SQL/DS Tables/Views	235
Incorporation Functions for Tables/Views	237

Compare - Code T	237
Connect - Code C	237
Display Masters - Code M	238
Incorporate - Code I	238
Incorporating DB2 Tables and Views in Batch Mode	239
Incorporating NDBs for IMS Databases/Segments	240
Incorporating NDBs for IMS Databases/Segments	240
Selecting Natural DBDs for IMS Databases and Segments	240
Incorporating Natural DBDs for IMS Databases and Segments	241
Incorporate - Code I	241
Replace - Code R	242
Incorporating IMS Database in Batch Mode	243
Incorporating Adabas Tables and Views	245
Incorporating Adabas Tables and Views	245
Prerequisites	245
Selecting Adabas Tables/Views	245
Incorporation Functions for Tables/Views	246
Compare - Code T	246
Connect - Code C	247
Display Masters - Code M	247
Incorporate - Code I	247
Incorporating Adabas Tables and Views in Batch Mode	247
Incorporating Natural DDMs	249
Incorporating Natural DDMs	249
Prerequisites and Restrictions	249
Selecting DDMs	249
Incorporation Functions for DDMs	250
Compare - Code T	250
Connect - Code C	251
Display Related DDMs - Code V	251
Incorporate - Code I	251
Incorporate DDM in Batch Mode	252
Incorporating COBOL Copy Code	254
Incorporating COBOL Copy Code	254
Prerequisites	254
Calling the Function	254
Incorporating COBOL Copy Code in Batch Mode	257
Representation of COBOL Field Definitions in Predict	257
Incorporating Super Natural Users	259
Incorporating Super Natural Users	259
Incorporating Super Natural User in Batch Mode	259
Incorporating Natural Security Users	261
Incorporating Natural Security Users	261
Prerequisites and Restrictions	261
Selecting Natural Security Users	261
Incorporating Natural Security Users in Batch Mode	262
Incorporating Tables / Views of SQL Database Systems	263
Incorporating Tables / Views of SQL Database Systems	263
Functional Scope	263
Prerequisites	263
Restrictions	263
Selecting SQL Tables and Views	263
Incorporation Functions for SQL Tables and Views	265
Display Masters - Code M	265
Incorporate - Code I	265
Incorporate SQL Tables / Views in Batch Mode	266

Comparison	267
Comparison	267
Concepts of Comparison	268
Concepts of Comparison	268
Functional Scope	268
Resetting the Modified after generation Flag	268
Format of Reports Created by Comparison Functions	269
Calling Comparison Functions	269
General Prerequisites for Comparison Functions	269
Selecting Objects for Comparison Functions	270
Check, List, Update and Save Options	270
Saving the Result of Comparison Functions in Sets	270
Calling Comparison Functions with Commands	270
Comparing Different Types of Objects	272
Comparing Different Types of Objects	272
Comparing Adabas Databases	272
Calling the Function	272
Comparing Adabas Databases in Batch Mode	273
Comparing Adabas Files	275
Comparing Adabas Files	275
Prerequisites and Restrictions	275
Calling the Function	275
Comparing Adabas Files in Batch Mode	278
Example	279
Comparing Vista Translation Tables	280
Comparing Vista Translation Tables	280
Prerequisites	280
Calling the Function	280
Comparing Vista Translation Tables in Batch Mode	280
Example	281
Comparing DDMs	282
Comparing DDMs	282
Prerequisites	282
Calling the Function	282
Comparing DDMs in Batch Mode	284
Example	284
Comparing Adabas Tables/Views	285
Comparing Adabas Tables/Views	285
Prerequisites	285
Calling the Function	285
Comparing Adabas Tables and Views in Batch Mode	287
Example	287
Comparing DB2 Databases	289
Comparing DB2 Databases	289
Prerequisites	289
Calling the Function	289
Comparing DB2 Databases in Batch Mode	290
Example	291
Comparing DB2 Storagegroups	292
Comparing DB2 Storagegroups	292
Prerequisites	292
Calling the Function	292
Comparing DB2 Storagegroups in Batch Mode	293
Example	293
Comparing DB2 Tablespace	294
Comparing DB2 Tablespace	294

Prerequisites	294
Calling the Function	294
Comparing DB2 Tablespaces in Batch Mode	295
Example	296
Comparing DB2 Tables and Views	297
Comparing DB2 Tables and Views	297
Prerequisites and Restrictions	297
Calling the Function	297
Comparing DB2 Tables and Views in Batch Mode	299
Example	300
Administration of External Objects	301
Administration of External Objects	301
Overview of Options	301
Overview of Administration Functions	301
General Rules	302
Commands	302
Batch Mode	303
Administrating External Objects Stored with Entire System Server	303
External Object Types and Administration Functions	303
Administrating Different Types of External Objects	305
Databases	305
Functions	306
Dataspaces	307
Functions	308
Files	309
Functions	310
Programs	312
Functions	314
Storagespaces	315
Functions	316
Preprocessor	317
Preprocessor	317
Overview	317
Using the Preprocessor	318
Using the Preprocessor	318
Calling the Preprocessor	318
Prerequisites - Work Files	319
Reports - Print Files	319
Preprocessor Statements	320
Preprocessor Statements	320
Overview of Preprocessor Statements	320
Statements to Include Copy Code Generated from Predict file objects	320
Statements to Write XRef Data for 3GL Copy/Include Code or Function Calls	320
Format of Preprocessor Statements	320
Using Keyword and/or Positional Parameters	321
CALL	321
COPY	321
Syntax	321
ENTRY	322
FORMAT-BUFFER	323
GENERATE	324
Generating Assembler Copy Code	325
Generating COBOL Copy Code	329
Generating PL/I Include Code	333
PROGRAM	337

- Creation of XRef Data 338**
- Creation of XRef Data 338
- What Types of Code Use are Documented in XRef Data 338
- Documenting CALL Statements and Entry Points 338
- Documenting the Use of Copy Code 338
- Documenting EXEC CICS Statements 339

External Objects in Predict - Overview

Many Predict functions process Predict documentation data as well as implemented code. This simplifies creation and maintenance of documentation and external objects and guarantees consistency between documentation and implementation.

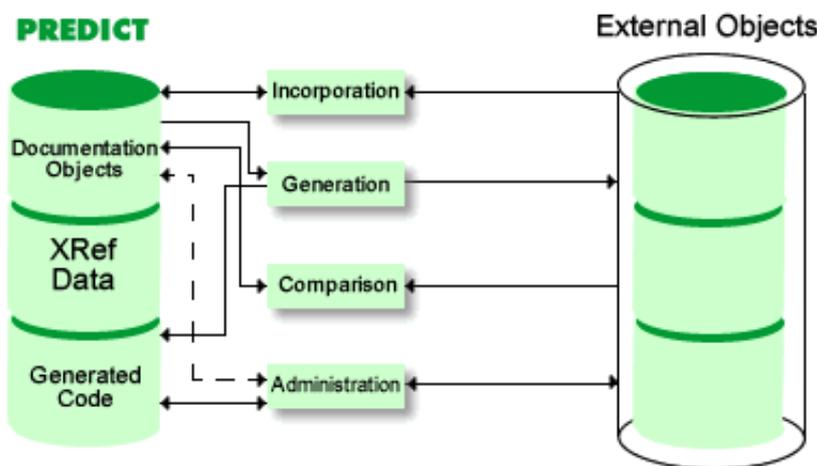
This document describes in detail the functions which process both documentation and external objects. It covers the following topics:

- Handling of External and Documentation Objects An overview of the functions which process documentation and external objects. The information provided in this section is essential for understanding many functions described in later sections of this document.
- Generation A wide variety of external objects can be generated from documentation objects stored in Predict.
- Working with File Implementation Plans Implementation plans are used to simplify the generation of external objects.
- Incorporation Predict objects can be created by incorporating information from implemented objects which are stored in an external environment.
- Comparison Predict objects and external objects can be compared, and if any inconsistencies are detected, either the documentation or external object can be updated.
- Administration of External Objects Implemented objects that are connected to Predict objects can be administered using Predict functions.
- Preprocessor The Preprocessor inserts copy code generated from Predict file objects into third generation language programs and writes XRef data which documents the use of these definitions and the calls of external members.

Handling of External and Documentation Objects

Information stored in Predict objects can be used to generate external objects, and documentation objects can be incorporated from external objects. The concepts of handling external and documentation objects in Predict are described in this section.

Information provided in this section is needed to understand many options of generation, incorporation and comparison functions.



This section covers the following topics:

- What is an External Object
Different types of external objects can be connected to Predict documentation objects. This section provides an overview and describes the general characteristics of different external object types.
- Connection of External and Documentation Objects
Predict ensures consistency of documentation and application by protecting external and/or documentation objects that are connected from being deleted or modified arbitrarily.
The connection between external and documentation objects is established by information stored with Predict objects.
- Overview of External Objects
This table shows which external objects can be processed with which functions.

What is an External Object

In Predict, data definition objects for use in applications that can be connected to Predict documentation objects are called **external objects**. There are basically two types of external objects:

- external objects owned by Predict (usually stored in the FDIC file)
- external objects **not** owned by Predict (usually stored in the external environment).

Note:

Special rules apply to DDMs and Natural processing rules. See the respective sections of section Generation in this documentation for more information.

External Objects Owned by Predict

The following types of external objects are owned by Predict:

- 3GL copy/include code (C, COBOL, Assembler, FORTRAN, PL/I)
- Adabas invert, compression and security definitions (ADAINV, ADAWAN/ADACMP/ADAFDU, ADASCR)
- Adabas/VSAM Bridge transparency table
- SQL CREATE statements

General Rules

The following rules apply to external objects that are owned by Predict.

Administration

- Objects of all the above types are generated from Predict file objects of the respective types.
- Up to 30 external objects per file and per language can be generated.
- The objects can be administered exclusively with Predict functions.
- The objects are dependent on the Predict documentation objects from which they were generated: if the documentation object is deleted, the generated objects are deleted as well.

Use

- The external objects of the above types are used at compile time.
- Copy code for use in 3GL programs must be copied (punched) to an operating system library before it can be used. Copying can be performed by Entire System Server. Storage of generated data definition objects as operating system members is possible in an OS/390 and VSE environment.
- The preprocessor can be used to generate 3GL copy code and include copy code into 3GL programs.

External Objects Not Owned by Predict

The following types of external objects are not owned by Predict: they belong to an application (development) environment.

- Databases (Adabas, DB2, IMS/DL1)
- Vista translation table
- DB2 tablespaces and storagegroups
- Files, tables and views (Adabas, DB2, SQL/DS, Adabas SQL Server)
- IMS User Defined Fields (UDFs)
- Natural DDMs (including Natural security definitions and/or Super Natural files)

General Rules

The following rules apply to external objects that are **not** owned by Predict.

Administration

- External objects of all these types can be generated from Predict documentation or be processed with Incorporate functions.
- Each external object can be connected to a documentation object with a generation or incorporation function.
For the impact of connecting external and documentation objects see Connection of External and documentation Objects.
- For external objects implemented with SQL (DB2, Adabas SQL Server, SQL/DS storagegroups, databases, tablespaces, dbspaces, tables and views) Predict stores the SQL statements that have been generated in a

generation protocol.

If several generation runs are executed, the protocol is extended for each generation.

- The objects can be administered with utilities of the application environment (for example SYSDDM, SYSAOS, SYSDB2, Natural map editor).

Use

- Most of the object types are used at run time. Some of these objects are stored directly in the application environment, others are stored in the Predict system file. For objects that are stored in the Predict system file, Predict data must therefore be accessible at run time of the application which uses the external objects.

Connection of External and Documentation Objects

Predict connects external objects and documentation objects if an external object has been generated from a documentation object or - vice versa - a documentation object has been incorporated from an external object.

Connecting external and documentation objects helps ensure the consistency of the documentation and an application: documentation objects and - to a certain extent - external objects that are connected are protected from being deleted or modified.

External objects owned by Predict need not be connected: because these types of objects can only be administrated with Predict functions, the consistency with documentation objects is not endangered.

Disconnecting Objects

External objects and documentation objects can be disconnected with Administration functions Disconnect Implementation and Purge Implementation. See the section Administration of External Objects in this documentation.

Impact of Connecting External and Documentation Objects

The connection of external objects and documentation objects affects the following activities.

- **Administration of External Objects**
The administration of external objects connected to documentation objects can be restricted with the Predict parameters of the SYSDDM utility, AOS, SYSDB2 utility and Rule in map editor. See Protection in the section **Defaults** in the **Predict Administration documentation**.
If these parameters are set to C (connected) the respective utilities cannot be used to administrate external objects that are connected to a Predict object.
- **Modification of Predict Objects**
Modification of Predict objects is affected as follows:
 - Predict file objects of type Adabas that are connected to an implemented file cannot be unlinked from the Predict database object they belong to.
 - The type of a file connected to an external object cannot be changed.
- **Purging Predict Objects**
Purging Predict objects is affected as follows:
 - Predict File and Verification objects connected to an external object cannot be deleted. To delete a Predict object connected to an external object, the two objects must be disconnected.
 - A generation protocol created by a Generate DB2 ... function can only be purged by purging or disconnecting the implemented DB2 object.
- **Incorporation of External Objects**
As a general rule, only external objects not yet connected to a documentation object can be processed with incorporation functions. However, for IMS and DL1 databases, a Replace option is available with which connected documentation objects can be overwritten.
- **Comparison of External and documentation Objects**

To compare an external object and a documentation object, both have to be connected.

Overview of External Objects

The table below provides a full list of all external object types supported by Predict.

Object	Code	Command	Generate	Incorporate	Compare	Administrate
Adabas Compression Definition	AC	WAN, CMP	Y			Y
Adabas Database	AD	ADABAS-DATABASE		Y	Y	Y
Adabas File	AF	FDT	Y	Y	Y	Y
Adabas Invert Definitions	AI	ADAINV	Y			Y
Adabas Security Definitions	AS	SCR	Y			Y
Vista Table	AT	VISTATAB	Y		Y	Y
Transp. Table for Adabas VSAM Bridge	AV	AVB ADAVSAM	Y			Y
Assembler Copy Code	BA	BAL ASSEMBLER	Y			Y
Adabas D Table / View	BF	ESD-TABLE		Y		
C Include Code	CC	LANG-C	Y			Y
COBOL Copy Code	CO	COBOL	Y	Y		Y
SQL CREATE Statement	CR	SQL-CREATE	Y			Y
Data Definition Module	DD	DDM	Y	Y	Y	Y
DB2 Database	D2	DB2-DATABASE	Y	Y	Y	Y
Adabas Table/View	EQ	ESQ	Y	Y	Y	Y
Fortran Copy Code	FO	FORTRAN	Y			Y
Ingres Table/View	JF	INGRES-TABLE		Y		
IMS Database	ND	NDB		Y		Y
Natural Security User	NS	Security		Y		
Oracle Table/View	OF	ORACLE-TABLE		Y		
PL/I Include Code	PL	PLI	Y			Y
Processing Rule	RU	RULE	Y	(Y)	(Y)	Y
DB2 Storagegroup	SG	STORAGEGROUP	Y	Y	Y	Y
Super Natural User	SU	SUPER		Y		
DB2 Table/View, SQL/DS Table/View	T2	TABLE	Y	Y	Y	Y
DB2 Tablespace, SQL/DS DBspace	TS	TABLESPACE	Y	Y	Y	Y
User-defined Fields for IMS	UD	UDF	Y			Y
Informix Table/View	XF	INFORMIX-TABLE		Y		
Sybase Table/View	YF	SYBASE-TABLE		Y		

Note for items marked with (Y) for SQL objects only:

If a check expression exists in the SQL catalog, the check expression can be incorporated or compared.

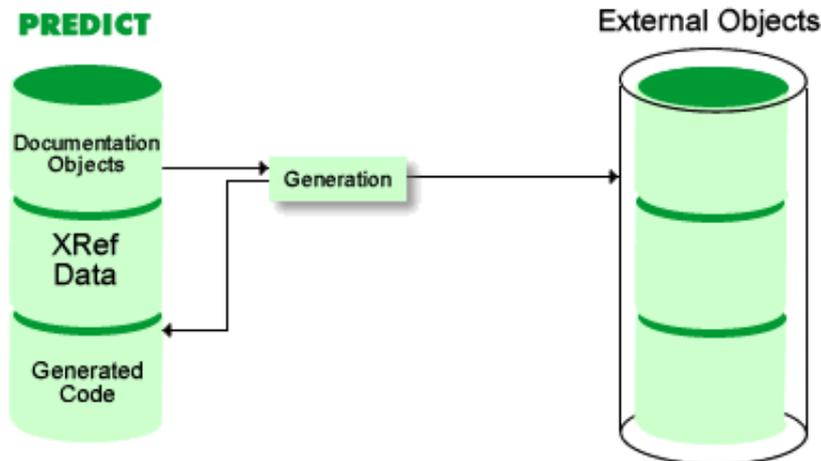
Generation of External Objects

This document covers the following topics:

- Concepts of Generation
- Calling Generation Functions
- Generation Defaults
- Storage of External Objects Owned by Predict
- Common Parameters
- Generate for Adabas
- Generate for Natural
- Generate for DB2
- Generating for SQL
- Generating for OS/400
- Generate for Third Generation Languages - 3GL
- Sample File Definition

Generation

External objects can be generated from Predict objects.



There are basically two types of external objects:

- external objects owned by Predict (usually stored in the FDIC file)
- external objects **not** owned by Predict (usually stored in the external environment).

Note:

Special rules apply to DDMs and Natural processing rules. See the respective sections of this section for more information.

This section covers the following topics:

- Concepts of Generation
- Calling Generation Functions
- Generation Defaults
- Storage of External Objects Owned by Predict
- Common Parameters

Concepts of Generation

Data definitions generated from Predict objects are called **external objects**.

Two types of external objects can be generated:

- external objects owned by Predict
- external objects not owned by Predict.

External Objects Owned by Predict

- 3GL copy/include code (C, COBOL, Assembler, FORTRAN, PL/I)
- Adabas invert, compression and security definitions (ADAINV/ADAWAN/ADAFDU, ADACMP, ADASCR)
- SQL CREATE statements
- Adabas/VSAM transparency table

External objects owned by Predict are used at compile time and have to be copied (punched) to an operating system library before they can be used in the implementation of an application. Copying can be performed by Entire System Server or the Preprocessor.

A variety of parameters determine how these types of external objects are stored. See Parameters Specifying the Form of Output.

External Objects not Owned by Predict

- Databases (Adabas, DB2, IMS/DL1)
- Vista translation table
- DB2 tablespaces and storagegroups
- Files, tables and views (Adabas, DB2)
- IMS User-Defined Fields (UDFs)
- Natural DDMs (including Natural security definitions and/or Super Natural files).

External objects not owned by Predict are used at run time. Some of these objects are stored directly in the application environment, others are stored in the Predict system file. For objects that are stored in the Predict system file, Predict data must be accessible when an application using the external objects is running.

Connecting External Objects and Documentation Objects

An external object that has been generated from a documentation object is regarded as connected to the documentation object. The connection is established by adding an implementation pointer to the documentation object. External objects and documentation objects that are connected are - to a certain extent - protected from being modified independently from each other.

See also section Handling of External and documentation Objects in this documentation.

Overview of Generation Functions

The following table is sorted by the code used to call a generation function from the Predict main menu.

Object	Code	Command	Description of External Object	Predict owned
Adabas compression definition	AC	WAN, CMP	For use as input to the Adabas compression utility ADACMP, ADAWAN or ADAFDU. If generating for ADACMP, you can generate additional input for ADALOD.	Y
Adabas File	AF	FDT	Adabas files are loaded directly into an Adabas database. If the file is already loaded, the differences of the implemented file and the documented file are determined and update commands are generated in order to transform the implemented Adabas file according to the documented file.	
Adabas invert definitions	AI	ADAINV	For use as input to the Adabas invert utility ADAINV.	Y
Adabas security definitions	AS	SCR	For use as input to the Adabas security utility ADASCR.	Y

Vista table	AT	VISTATAB	This function implements Vista elements of Predict file and database objects of type A in the translation tables of Vista.	
Transparency table for Adabas VSAM bridge	AV	AVB, ADAVSAM	Supports conversion from VSAM files with KSDS organization to Adabas files without the need to change existing COBOL programs.	Y
Assembler Copy Code	BA	BAL, ASSEMBLER	Copy code for use in a 370 Assembler program.	Y
C Include Code	CC	LANG-C	Include code for use in a C program.	Y
COBOL Copy Code	CO	COBOL	Copy code for use in a COBOL program.	Y
SQL CREATE statement	CR	SQL-CREATE	CREATE TABLE, CREATE VIEW or CREATE CLUSTER statement. These SQL statements are stored as Natural members.	Y
Data Definition Module	DD	DDM	A collection of field definitions used by Natural for accessing a database.	
DB2 database	D2	DB2-DATABASE	A DB2 database is implemented directly as a physical DB2 database. Not applicable to SQL/DS.	
Adabas table/view	EQ	ESQ	Table descriptions or views in an Adabas SQL Server catalog.	
FORTRAN Copy Code	FO	FORTRAN	Copy code for use in a FORTRAN program.	Y
OS/400 file	O4	GENOS4	OS/400 file definitions.	
PL/I Include Code	PL	PLI	Include code for use in a PL/I program.	Y
DB2 procedure/function	P2	DB2-PROCEDURE	This function requires an object of type Program as input, from which then either a procedure or a function is generated.	
Verification rule	RU	RULE	A rule must already have been generated using the Generate DDM function. Only the code of the rule is changed. The new rule will automatically be used by Natural maps that are cataloged. Not applicable to rules of status SQL or Natural Construct.	
DB2-Storagegroup	SG	STORAGEGROUP	Storagegroups are implemented directly from Predict storagespaces. If a storagegroup has already been implemented from the storagespace, the differences of the implemented DB2 storagegroup and the documented storagespace are determined and update commands are generated to adapt the implementation to the documentation.	

DB2 table/view	T2	TABLE	DB tables/views are implemented directly from Predict file objects of type D or E. If a table/view has already been implemented from the Predict table/view definition, the differences of the implemented DB2 table/view and the Predict table/view definition are determined and update commands are generated in order to change the implementation according to the documentation.	
DB2 tablespace, SQL/DS DBspace	TS	TABLESPACE	Tablespaces/dbspaces are implemented directly from Predict dataspace. If a tablespace/dbspace has already been implemented from the dataspace, the differences of the implemented DB2 tablespace/ SQL/DS DBspace and the documented dataspace are determined and update commands are generated to transform the implementation according to the documentation.	
User-defined fields for IMS	UD	UDF	Definitions used by Natural to access an IMS database.	

Calling Generation Functions

Generation functions are called with G in the field Function and the appropriate code in the field Object Type in a Predict main menu or with a command. The following rules apply:

- All generation functions can be executed both online and in batch mode.
- If a system is not installed, the respective generation functions are not available. For example, if DB2 is not installed at your site, no generation functions for DB2 objects are available.

Calling Generation Functions with Commands

- Parameters are entered on the line following the command.
- Parameters can be entered in positional and/or keyword form. See the section Predict Commands in the **Predict Reference documentation**.
- To use the default parameters defined for the function, enter the command and the <object-ID> on the same line. This does not apply in batch mode.
- Some generation functions can be called with two or three commands. It makes no difference which command is used.

Generation Condition Codes

Condition codes are described in the section Predict in Batch Mode in the **Predict Administration documentation**.

Generation Defaults

Default generation values are set at installation. Most default values are displayed in the input screen of the respective generation function and can then be overwritten for temporary use. Changes to default values apply to subsequent generation runs until another Predict function is executed. This does not apply in batch mode.

The following rules apply for the use of default values for generation parameters:

- Default values of generation parameters can be changed with the function Generation Defaults in the Modify Defaults Menu or the command DEFAULT object code. For example: DEFAULT COBOL displays the Modify COBOL Defaults screen shown below.

```

13:35:47          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Modify COBOL Defaults -
                                Modified 2003-05-31 at 13:31
                                by CHD

Mark with 'X' the options which may be modified by the user.

X Save as member .....
X Overwrite option ..... Y (Y,N)
X Punch / output .....* N
X List generated code ..... Y (Y,N)
X Generate format buffer ..* N
X Check field names .....* Y
X Start level ..... 1 (0-40)
X Level number increment.... 1 (1-40)
X Level shift increment ... 3 (0-9)
X Nr. of abstract lines ... 5 (0-16)
X Generate initial value ..* N
X Synchronized .....* Y
X Depending on ..... N (Y,N)
X Record buffer name .....
X Format buffer name .....

X Save in library .... COBLIB
X Op. system member ..
X List offsets .....* N
X Adabas version ....* I7
X Field name prefix ..
X Field name suffix ..
X Validate ..... -
X Truncation .....* R
X With Cond. names ... Y (Y,N)
X Indexed by .....* N
X Literal delimiter .* S
X Decimal character .* P
X Redefinition name .* S

Preprocessor force ..... N (Y,N)
                                Compiler .....* 7
                                Library system ..... 2

```

- Generation defaults can be protected by blanking out the X preceding the parameter in Modify ... Defaults screens. Protected default values cannot be changed when executing a generation function. The input fields of protected parameters are locked in the input screen of the generation function. These fields are skipped when positioning the cursor with the TAB key.
- Some default values are not displayed in the input screen of a generation function and can therefore only be changed using the Modify Generation Defaults function. These parameters are described under **Presetting** in the descriptions of individual generation functions later in this section.
- For Predict-owned external object types you can specify default values for storing generated code. See Specifying Entire System Server and VSE Librarian Options.

For a description of the parameters shown in the previous screen, see the relevant generation functions later in this documentation, the parameter Preprocessor force is described below.

Preprocessor Force

This parameter is used by the Predict Preprocessor and Adabas Native SQL. It can be specified for Assembler, COBOL, FORTRAN, PL/I, and ADA. It can only be defined in the respective Modify Generation Defaults screen.

Valid values:

Y

Both the Predict Preprocessor and Adabas Native SQL check that the program to be processed is documented. If no Predict object documenting the program is found, the task is not executed and a message is returned.

N

No check is performed. Default setting when Predict is installed.

Storage of External Objects Owned by Predict

There are different options to output/store external objects owned by Predict. The following types of external objects are owned by Predict:

- 3GL copy/include code (C, COBOL, Assembler, FORTRAN, PL/I)
- Adabas invert, compression and security definitions (ADAINV, ADAWAN/ADAFDU/ADACMP, ADASCR)
- Adabas/VSAM transparency table
- SQL CREATE statements.

Output / Storage Options for External Objects

Option 1 - Generating External Objects Temporarily

This option is used to have a first look at the results of a generation function. To generate objects temporarily the parameter Save as member is left blank. The parameter List generated code determines what is displayed.

Option 2 - Storing Generated External Objects as Members in a Natural Library of the Predict System File

If the parameter Member name is specified, the generated external objects are stored in a Natural library in the Predict system file (FDIC). The library must be specified.

For each file you can save up to 30 members per language.

For DB2 objects (these are not owned by Predict) a generation protocol can be created and stored in Predict by specifying the parameters Protocol saved in member and Protocol saved in library.

Members in a Natural library that have been created with generation functions can be read by the Preprocessor or written to workfile 1 using the command PUNCH or WRITE. See description of PUNCH/WRITE command in the section Predict Commands in the **Predict Reference documentation** and section The Preprocessor in this documentation.

Option 3 - Storing Generated External Objects Additionally in a Workfile or - with Entire System Server - as an Operating System Member

External objects can additionally be stored by Entire System Server as operating system members. The following rules apply:

- Storage by Entire System Server can only be executed in addition to storage in the Predict system file (Save as member parameter must be specified).
- Storage of generated external objects as operating system members is only possible in an OS/390 or VSE environment.
- If Library system is set to 3, the external object is stored additionally in a VSE Librarian library.
- One operating system member can be stored for each member in a Natural library that has been generated by a generation function.
- External objects are not processed by Entire System Server if errors occur during the generation.

Note:

See also Storing External Objects with Entire System Server.

Parameters Specifying the Form of Output

Parameters specifying the form of output of external objects owned by Predict are described below. Each parameter applies to different data storage options.

Parameters	
Save as member (Option 2)	<p>If this parameter is specified, the generated code is stored as a member with the given name in a Natural library on the Predict system file (FDIC). The first character of the member name must be alphabetic, all others must be alphanumeric.</p> <p>If you enter an asterisk, the name of the new member is determined by Predict as follows:</p> <ul style="list-style-type: none"> • If a Predict File object is used for the first time to generate a member, the member name is taken from the File ID (truncated if necessary). • If at least one member has been generated before, a selection window containing all member names from previous generation runs is displayed. Select an existing member name with X or enter X in the line *new* to specify a new member name to be created. <p>In batch mode, a new member name is created in either case. If 30 members already exist, the oldest is overwritten.</p>
Save in library (Option 2)	<p>The generated definitions are saved in the specified Natural library on the Predict system file. The library name must not start with SYS; its first character must be alphabetic and all others must be alphanumeric.</p> <p>This parameter must be specified if the generated external object is to be stored in a Natural library.</p>
Overwrite option (Option 2)	<p>Y</p> <p>An existing member that has been generated from the same file object by Predict is overwritten. Existing members that were not generated by Predict are not overwritten. If the generated external object has additionally been stored as an operating system member, this member is overwritten, too.</p> <p>If an external member was created and the generation function is executed a second time without entering an external member, the external member is deleted if Overwrite option=Y.</p>
Op. system member (Option 3)	<p>If the generated code is to be stored in an operating system member, a member name must be specified with this parameter and the options Punch/Output, Save as member and Save in library must be specified.</p> <p>If an operating system member name is not specified explicitly, a name can be given with one of the following methods:</p> <ul style="list-style-type: none"> • if an asterisk is entered, the first 8 characters of the file ID are used as the member ID. • given that an external object has been stored both as a member of a Natural library and as an operating system member in the same generation operation, and that Entire System Server is used, the field Op. system member can be left blank. Predict will insert the name of the existing operating system member into the input field. This name can then be changed.

<p>Punch/Output (Option 3)</p>	<p>Determines further use of the generated code.</p> <p>N The generated external object is neither processed by Entire System Server nor written to a sequential file (workfile).</p> <p>Y The generated external object is punched to a sequential file (workfile 1): If Op. system member (see above) is specified, operating system statements preceding the data definitions are created in the external object depending on the type of operating system specified with the parameter Library system further down this table. Option Y is only valid in online environments where access to workfiles is possible, (for example, BS2000, CMS, TSO), and when running in batch mode.</p> <p>If Library system=3 the external object is additionally stored in a VSE Librarian library. See Specifying Entire System Server and VSE Librarian Options.</p> <p>P Entire System Server stores the generated external object as an operating system member. The parameter Op.system member and Entire System Server options specified in the generation defaults determine where the object is stored.</p> <p>S As above, but where the object is stored can be specified in a subsequent screen. See Specifying Entire System Server and VSE Librarian Options.</p>
<p>List generated code (Option 1)</p>	<p>Determines the amount of information to be displayed.</p> <p>Y The generated code is displayed.</p> <p>N Only the messages and the summary of the messages is displayed.</p>

<p>Library system (Option 3)</p>	<p>This parameter can only be specified in the generation defaults. Determines which type of operating system statements preceding the data definitions in the external object are created.</p> <p>blank Code is generated in IEBUPDTE format.</p> <p>A-Z CATALS and BKEND cards are generated for VSE and the member is added to the sublibrary identified by the specified letter. These cards will also be generated if the Predict command PUNCH is used in batch mode.</p> <p>1 Cards for The Librarian are generated.</p> <p>2 PANVALET cards are generated.</p> <p>3 VSE SP2/SP3 cards for use with LIBRARIAN system are generated. The ACCESS and CATALOG statements with the options Library name, Sublibrary name and Member type are added. If Punch/output is set to Y, the external object is stored additionally in a VSE Librarian library.</p> <p>The following table shows the statements generated for the various operating systems:</p> <table border="1" data-bbox="411 1115 1018 1413"> <thead> <tr> <th>Operating System</th> <th>Statement</th> </tr> </thead> <tbody> <tr> <td>OS</td> <td>IEBUPDTE ADD</td> </tr> <tr> <td>VSE</td> <td>CATALS, BKEND</td> </tr> <tr> <td>OS Librarian</td> <td>ADD</td> </tr> <tr> <td>Panvalet</td> <td>ADD</td> </tr> <tr> <td>DOS Librarian (VSE SP2/SP3)</td> <td>ACCESS,CATALOG</td> </tr> </tbody> </table> <p>Note: This parameter is evaluated only if the Punch/Output option is set to Y and an Op. system member is specified. This parameter is not applicable when generating ADAFDU definitions with function Generate ADACMP/ ADAWAN definitions with the following Adabas versions:</p> <ul style="list-style-type: none"> ● any version starting with U, ● V3 and above or ● P2 and above. <p>A complete table of Adabas versions can be found at the end of this section.</p>	Operating System	Statement	OS	IEBUPDTE ADD	VSE	CATALS, BKEND	OS Librarian	ADD	Panvalet	ADD	DOS Librarian (VSE SP2/SP3)	ACCESS,CATALOG
Operating System	Statement												
OS	IEBUPDTE ADD												
VSE	CATALS, BKEND												
OS Librarian	ADD												
Panvalet	ADD												
DOS Librarian (VSE SP2/SP3)	ACCESS,CATALOG												
<p>Workfile name</p>	<p>Only for Windows or UNIX platforms. Identifies the file for punch output. If punch is set to Y, default is taken from Natural Parameter Module -> Workfile name 1.</p>												

Storing External Objects with Entire System Server

External Objects owned by Predict can be stored directly as a member in an operating system library (partitioned data set) in an OS/390 and a VSE environment if Entire System Server is available.

This option is used by setting the parameter Punch/Output to P or S.

If Punch/Output is set to S additional parameters can be specified in a subsequent screen. See Specifying Entire

System Server and VSE Librarian Options.

Additional storage of generated code with Entire System Server is notified in the generation log of the Predict file object from which the code was generated.

The following rules apply:

- **In a VSE environment**

Members are identified by member name and member type. Members of different types can therefore have the same name.

The code is stored as member in a VSE library.

The following prerequisites must be met:

- Librarian utility must be installed (VSE/SP2 or higher)
- library and sublibrary must be defined in VSE
- member type must be specified.

- **In an OS/390 environment**

Members are identified by name only.

The code is stored as a member in a partitioned data set.

The data set must be allocated. It is recommended that the data sets are cataloged. In this case, the VOLUME name need not be specified.

Note:

Parameters that have to be specified when storing generated code with Entire System Server are described in the section Specifying Entire System Server and VSE Librarian Options.

Using Entire System Server in Networks

Generated code can be stored in operating system libraries residing in remote environments. In this case Entire System Server calls are distributed by Software AG's Entire Net-Work facility.

Prerequisites for Using Entire System Server

- Depending on the setup parameters specified during installation of Entire System Server a LOGON is required.
 - Online, the LOGON is requested if necessary.
 - In batch mode, specify a logon command before the first Entire System Server command:

```
NPRLOGON <userid>,<password>,<Entire System Server database-id>
```

for example:

```
NPRLOGON AZ,XYZ,ENTIRE-SYSTEM-SERVER
```

To prevent the password being displayed in SYSOUT in batch mode, enter the terminal command %* before NPRLOGON.

- The database number 148 must be defined as Entire System Server in the Natural Parameter module by the NTDB macro (NTDB PROCESS,148).
The Entire System Server node can have a number other than 148, because the node number is always specified when an Entire System Server access is performed in Predict. In this case there must be an additional PROCESS node definition in the Natural parameter module.
- Predict supports multiple Entire System Server nodes. If the generated code is to be written by Entire System Server, the Entire System Server database ID must always be defined and the node number is filled with the logical database number.

Regenerating Code Handled by Entire System Server

When regenerating code that has been processed by Entire System Server, the members stored in a library or partitioned data set must be updated to ensure consistency.

Code can be regenerated with either the same or with different Entire System Server options:

- When regenerating code without changing the operating system member name, the Entire System Server options used for the previous generation are inserted in the generation screen (under the prerequisite that the options are valid according to the relevant defaults).
See Specifying Entire System Server and VSE Librarian Options.
- To regenerate code with different Entire System Server options, set the Punch/Output option to S. The options can then be changed in the second screen.

Example: A new Op. system member is specified. As the result the old operating system member is deleted and a new member is created.

If code is regenerated for which an operating system member was generated in a previous generation, and the Punch/Output option is set to N or Y, this will cause the operating system member to be deleted after a warning has been issued. The contents of the member will not be compared.

If code is regenerated and the operating system member documented in the generation log of the Predict file object is not found (this means that the member was deleted by an operating system utility) the code can be written to a new member or the generation log of the file is deleted depending on the Punch/Output option.

If Entire System Server is deinstalled, the generated code can be regenerated and the link in the generation log of the file is deleted.

If Entire System Server is temporarily not active (example: response code 5999), the generation function for this member is rejected. If it is necessary under these circumstances to change this generated code (saved with the current member and library name), it must be deleted with the Predict function Administration Implemented File.

The table below shows the effects of different generation settings when working with Entire System Server:

Member exists see ¹⁾	Overwrite Option	Code written with Entire System Server see ²⁾	Opsys Member exists	Name of Opsys Member changed	Entire System Server available	Action
-	-	Y	N	-	Y	Write to new opsys member
Y	Y	Y	Y	N	Y	Opsys member replaced
Y	Y	Y	Y	Y	Y	Old opsys member deleted; write to new opsys member
Y	Y	N	Y	-	Y	Old opsys member deleted
Y	Y	Y	N (but documented)	-	Y	Write to new opsys member
Y	Y	N	N (but documented)	-	Y	Delete references in generation log to opsys member
Y	Y	N	Y	-	N (not installed)	Delete references in generation log to opsys member
Y	Y	N	Y	-	N (not active)	Generation function rejected

¹⁾ Field Save as member filled with valid value

²⁾ Parameter Punch/output = P or S

Specifying Entire System Server and VSE Librarian Options

Additional parameters must be specified if generated external objects are written to an operating system member with Entire System Server (Punch/Output=S) or are written directly as members in an VSE librarian library (Punch/Output=Y and Library system=3).

```

13:59:30          ***** P R E D I C T  4.3.1  *****                2003-05-31
                    - Punch / Output Default Options -
                                                Modified 2003-05-31 at 13:28
                                                by CHD
Mark with 'X' the options which may be modified by the user.

X Entire System Server Database ID ..* PROCESS-148

MVS Entire System Server Defaults
X Data set .....
X Volume .....

VSE LIBRARIAN Defaults
X Library .....
X Sublibrary .....
X Member type .....
X VSAM catalog name ..                ( Required for Entire System Server)

Previous entered default options
Op. system member ..
Library system ..... 2
Punch / output ..... N
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Parameters	
Entire System Server Database ID	ID of the Predict database object documenting the Entire System Server node. The logical database number specified in Predict is used as Entire System Server node number.
Data Set	Name of the PDS in an OS/390 environment used by Entire System Server.
Volume	Name of the Volume where the PDS used by Entire System Server is allocated. Volume must be specified if the data set is not cataloged in an OS/390 environment.
Library	Library name of the Librarian system (VSE SP2/SP3) or used by Entire System Server.
Sublibrary	Sublibrary name of the Librarian system (VSE SP2/SP3) or used by Entire System Server.
Member type	Member type of the Librarian system (VSE SP2/SP3) or used by Entire System Server.
VSAM catalog name	VSAM catalog name used only by Entire System Server.
Op. system member	The Op. system member name entered in the previous screen is displayed but cannot be modified in this screen.

Specify library system Librarian (VSE SP2/SP3) by setting option Library system to 3. Generated code written to workfile 1 is prepared for use as input of Librarian (VSE SP2/SP3) system: ACCESS and CATALOG statements are inserted at the beginning of the generated code.

Common Parameters

The parameters Adabas version and Preprocessor force can be specified with many generation functions. Preprocessor force can only be defined in the respective Modify Generation Defaults screen.

Adabas Version

Note:

If a new Adabas version is released that does not have any effect on Predict Generation functions, this new version will not appear in the selection menu. Use the code for the old version.

Code	Version	Remarks
I1	V 5.1 for IBM/Siemens	Applicable to all external object types for which this parameter can be specified. When generating copy/include code, sub/superdescriptors are not included in the record buffer layout.
I3	V 5.3 for IBM/Siemens	As above.
I6	V 6.1 for IBM/Siemens	As above. This Adabas version supports larger database and file numbers.
I7	V 7.1 for IBM/Siemens	As above.
O4	V 4.1 for IBM/Siemens	
U1	V 1.1 for UNIX	
U2	V 1.2 for UNIX	
U3	V 2.1 for UNIX	
U4	V 2.2 for UNIX	
U5	V 3.1 for UNIX/NT	
U6	V 3.2 for UNIX/NT	
V2	V 2.1 for VMS	
V3	V 3.1 for VMS	
V4	V 3.2 for VMS	
V5	V 4.1 for VMS	
P1	V 1.0 for OS/2.	
P2	V 1.2 for OS/2.	
R1	V 5.1 for IBM/Siemens	Only applicable to generation of copy/include code. Sub/superdescriptors are included physically in the record buffer layout. Code generated with this Adabas version cannot be used for update statements. Not applicable for files where parameter Adabas SQL usage =Y.
R3	V 5.3 for IBM/Siemens	As above.
R7	V 7.1 for IBM/Siemens	Similar to I7, sub/super and collation descriptors are included physically in the record buffer layout.

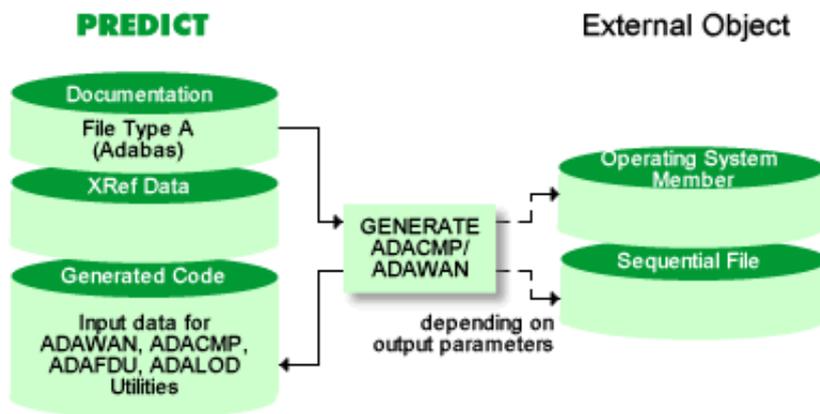
Generate For Adabas

The following types of external objects can be generated for Adabas:

- ADACMP, ADAWAN, ADAFDU and ADALOD Definitions
- Adabas File
- ADAINV Definitions
- ADASCR Definitions
- Vista Translation Table
- Adabas VSAM Bridge Transparency Table
- Adabas Table/Cluster/View

ADACMP, ADAWAN, ADAFDU and ADALOD Definitions

The function Generate ADACMP/ADAWAN Definitions generates input data for the Adabas compression / load utilities from Predict file objects of type A (Adabas files). The definition generated depends on the parameters Adabas version and Generate loader.



Note:

For more information see the section **Compression** or **ADACMP** in the **Adabas Utilities documentation**.

This section contains:

- Calling the Function
- Additional ADALOD Parameters
- Additional ADAFDU Parameters
- Additional Parameters For UES Support
- Generate ADACMP, ADAWAN, ADAFDU or ADALOD Definitions in Batch Mode
- Sample Output

Calling the Function

The Generate ADACMP/ADAWAN Definitions screen is displayed with function code G and object code AC in a Predict main menu, or with one of the commands GENERATE ADACMP or GENERATE ADAWAN.

```

13:04:49          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Generate ADACMP/ADAWAN Definitions -

File ID .....*

Phys. File number.....
Contained in DA .....
Phys. Database number ..

Save as member .....          Save in library .... WANLIB
Overwrite option ..... Y      (Y,N)          Op. system member ..
Punch / output .....* N
Generate loader ..... N      (Y,N)          UES ..... N (Y,N)
List generated code .... Y    (Y,N)          Adabas version ....* I7
List input file .....* Y
Input file ID .....
NUMREC .....          (-1=zero)          Cipher code .....
Occurrences used .....* Y
Record length .....          (*,number)          Record format .....*
Device statistics ..... Y    (Y,N)

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults
 The parameter below can be changed in the Modify ADAWAN/ADACMP Defaults screen.

Presetting	
Library system	Determines which type of operating system statements are generated in the external object before the data definitions. This parameter only applies if you set the parameter Punch/Output to Y and specify an operating system member under Op. system member.

The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output, Workfile name and List generated code are described in Parameters Specifying the Form of Output. The parameter Punch/output has an additional value for this external object type and is described below.

Parameters	
File ID, Phys. file number, Contained in DA, Phys. database number	The ADACMP, ADAWAN, ADAFDU and ADALOD definitions can be generated for one specific Adabas file in a database. This file is uniquely defined by the physical file number and the physical database number. This file can be identified by these four parameters. If the parameters are not unique, a selection screen appears. It is also possible to generate the definitions for an Adabas file which is not linked to any database. In this case the currently defined file attributes are taken.
Punch/Output	In addition to the standard options you can also specify the value D for this external object type: Download to PC (Write to workfile 7). For this setting, the following prerequisites must be met: <ul style="list-style-type: none"> ● Workfile 7 must be defined in the session variables as a PC file. ● Parameter Save as member must be specified

Adabas version	The version of Adabas for which the external object is to be generated. Enter an asterisk for valid values or see Adabas Version for more information.			
	This parameter determines the utility for which cards are generated. See table below.			
	Code	Version	Generated Card Format	Note
	I1	V 5.1 for IBM/Siemens	ADACMP	If parameter Generate loader is set to Y , ADALOD definitions are generated as well. See Additional ADALOD Parameters
	I2	V 5.2 for IBM /Siemens		
	I3	V 5.3 for IBM/Siemens		
	I6	V 6 for IBM/Siemens		
	I7	V 7.1 for IBM/Siemens		
	O4	V 4.1 for IBM/Siemens	ADAWAN	
	U1	V 1.1 for UNIX	ADAFDU	If parameter Generate loader is set to N, only field definitions are generated.
	U2	V 1.2 for UNIX		If Generate loader is set to Y, ADAFDU file definitions are generated in addition to the field definitions. See Additional ADAFDU Parameters
	U3	V 2.1 for UNIX		
	U4	V 2.2 for UNIX		
	U5	V 3.1 for UNIX/NT		
	V2	V 2.1 for VMS	ADACMP	Only field definitions are generated.
V3	V 3.1 for VMS	ADAFDU	If parameter Generate loader is set to N, only field definitions are generated.	
V4	V 3.2 for VMS		If Generate loader is set to Y, ADAFDU file definitions are generated in addition to the field definitions. See Additional ADAFDU Parameters	
V5	V 4.1 for VMS			
P1	V 1.0 for OS/2.	ADACMP		
P2	V 1.2 for OS/2.	ADAFDU		

Generate loader	<p>The system behavior depends on the card format to be generated. The card format in turn depends on the parameter Adabas version. See table above and Additional ADALOD Parameters and Additional ADAFDU Parameters.</p> <p>You can only set this parameter to Y for the following Adabas versions: I1, I2, I3, I6, I7, U1, U2, U3, U4, U5, V3, V4, V5.</p>
UES	<p>If you set this parameter to Y a window appears in which you can specify additional parameters concerning UES support. These parameters are described in detail in your Adabas documentation.</p> <p>You can only set this parameter to Y if the database for which the ADACMP cards are to be generated has the Predict attribute UES set to Y or if the file doesn't belong to any database.</p> <p>You can only set this parameter to Y for Adabas version I7. See Additional Parameters For UES Support for details.</p>
List input file	<p>This parameter is only evaluated if an input file is entered.</p> <p>Y The attributes of the fields of the input file and their corresponding Adabas fields are listed.</p> <p>N Attributes are listed only for incompatible fields.</p> <p>D All differences between the attributes of the input file and of the Adabas file are listed.</p>
Input file ID	<p>ID of the Predict file object from which the format buffer of the input file for the ADACMP utility is to be generated. In Adabas Version 5 and above, the compression utility need not use the standard format buffer layout based on the FDT: the order and format of the fields can be defined in a file specified by this parameter. This file must either be sequential (file type S) or a userview (File type U) of the Adabas file. A sequential file specifies fields using their field IDs; a userview specifies them using their field short names.</p> <p>Note: Only applies to Adabas Version 5 and above for IBM/Siemens.</p>
NUMREC	<p>A NUMREC parameter with the value provided is generated in the ADACMP control statement.</p> <p>If the value in this field is zero, it is ignored.</p> <p>If NUMREC = -1 the option NUMREC = 0 is inserted.</p> <p>Note: Only applies to Adabas Version 5 and above for IBM/Siemens.</p>
Cipher code	<p>Applies only to files marked as ciphered. The value of the cipher code parameter to be generated in the ADACMP control statement.</p> <p>Note: Only applies to Adabas Version 5 and above for IBM/Siemens.</p>

Occurrences used	<p>This parameter determines how many occurrences of periodic groups and multiple value fields are taken from the input record and used by the Adabas compression utility.</p> <p>Valid values:</p> <p>Y The number of occurrences defined in Predict is taken.</p> <p>M The maximum number of occurrences is always taken: for multiple value fields: 191 for periodic groups: 99 or 191 depending on the Adabas version.</p> <p>N No occurrences.</p> <p>Note: If you are generating for V 1.0 for OS/2 (Adabas Version=P1), this parameter must be set to Y. For Adabas on UNIX systems Occurrences used must be set to N.</p>
Record length	<p>A LRECL parameter with the value provided is generated in the ADACMP control statement.</p> <p>0, blank Field is ignored.</p> <p>*</p> <p>If an asterisk is entered, the record length of the input file is calculated for the LRECL parameter. This is only possible if the input record contains neither periodic groups with variable occurrences nor multiple value fields with variable occurrences.</p> <p>Note: Only applies to Adabas Version 5 and above for IBM/Siemens.</p>
Record format	<p>A RECFM parameter with the value provided is generated in the ADACMP/ADAWAN control statement.</p> <p>Valid values: F, FB, V, VB, U or blank.</p> <p>Note: Only applies to Adabas Version 5 and above for IBM/Siemens.</p>
Device statistics	<p>Y The device option is generated for the default data device type or for all data device types of the linked database.</p>

Additional ADALOD Parameters

If you set the parameter Generate loader to Y for Adabas Version I1, I2, I3, I6 or I7, a window appears in which you can specify additional parameters for the ADALOD utility. See your Adabas documentation for a detailed description of these parameters.

```

+-- Additional parameters for ADALOD -----+
!
!   Format DS/index .. Y (Y,N)                !
!   SORTDEV .....* 3380                      !
!   TEMPDEV .....* 3380                      !
!   SORTSIZE .....          25 C/B ... C      !
!   TEMPSIZE .....          30 C/B ... C      !
!   ISN pool size ....          KB/B .. KB    !
!   Work pool size ...          KB/B .. KB    !
!   SKIPREC .....                !
!   ETID .....                !
!   Force allocation . Y (Y,N)                !
+-----+

```

You can modify the default values in the window Additional parameters for Loader of the function Modify ADAWAN/ADACMP Defaults.

ADACMP definitions are generated first, followed by ADALOD definitions. See example below.

```

\\
ADALOD LOAD
ADALOD FILE=77
ADALOD NAME=CHD-A-FILE
ADALOD ACRABN=1
ADALOD ASSOPFAC=10
ADALOD DATAFRM=YES
:
:

```

The ADALOD utility must be called explicitly using the ADALOD definitions generated with this function.

Additional ADAFDU Parameters

If you set the parameter Generate loader to Y for Adabas version U1, U2, U3, U4, U5, V3, V4 or V5, a window appears in which you can specify additional parameters for the ADAFDU utility. These parameters are described in detail in your Adabas documentation.

```

+-- Additional parameters for ADAFDU -----+
!
!   Format DS/Index .. Y (Y,N)                !
!   Contiguous AC .... Y (Y,N)               !
!   Contiguous DS .... Y (Y,N)               !
!   Contiguous NI .... N (Y,N)               !
!   Contiguous UI .... N (Y,N)               !
+-----+

```

You can modify the default values in the window Additional parameters for Loader of function Modify ADAWAN/ADACMP Defaults.

Additional Parameters For UES Support

If you set the parameter UES to Y for Adabas Version I7 or above, a window appears in which you can specify additional parameters for the ADACMP utility.

Field	Keyword	Position
File ID	FILE-ID	1
Save as member	MEM	2
Save in library	LIB	3
Overwrite option	REPLACE	4
Op. system member	OS-MEMBER	5
NUMREC	NUMREC	6
Record length	LRECL	7
Record format	RECFM	8
List generated code	LIST	9
Punch / output	PUNCH	10
Occurrences used	USE-OCC	11
Input file ID	INFILE-ID	12
List input file	LIST-IN	13
Cipher code	CIPHER	14
Adabas version	ADA-VER	15
Device statistics	DEVICE	16
Phys. file number	FNR	17
Database ID	DB	18
Phys. database number	DBNR	19
Workfile name (see note below)	WORKFILE-NAME	20
If Entire System Server is used		
- DB-ID	NP-DBID	21
- Dataset	NP-DSNAME	22
- Volume	NP-VOLSER	23
- Library	NP-LIB	24
- Sublibrary	NP-SUBLIB	25
- Member type	NP-MEMTYPE	26
- VSAM catalog	NP-VSAMCAT	27

Note:

You can not specify a value other than blank (' ') for WORKFILE-NAME on mainframes.

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

Field	Keyword	Position
Generate Loader	FILE-ATTRIBUTES	28
Additional ADAFDU parameters		
Format DS/Index	FORMAT-DS-IND	29*
Contiguous AC	CONTIG-AC	30
Contiguous DS	CONTIG-DS	31
Contiguous NI	CONTIG-NI	32
Contiguous UI	CONTIG-UI	33
Additional ADALOD parameters		
Format DS/index	FORMAT-DS-IND	29*
SORTDEV	SORTDEV	34
TEMPDEV	TEMPDEV	35
SORTSIZE	SORTSIZE	36
SORTSIZE C/B	SORTSIZE-T	37
TEMPSIZE	TEMPSIZE	38
TEMPSIZE C/B	TEMPSIZE-T	39
ISN pool size	LIP	40
ISN pool size KB/B	LIP-T	41
Work pool size	LWP	42
Work pool size KB/B	LWP-T	43
SKIPREC	SKIPREC	44
ETID	ETID	45

* Parameter FORMAT-DS-IND may be used for ADAFDU as well as ADALOD.

Field	Keyword	Position
Additional UES parameters		
UES	GEN-UES	46
Byte order	BYTE-ORDER	47
Encoding family	ENC-FAMILY	48
Floating point format	FLOATING	49
UA code	UACODE	50
UW code	UWCODE	51

Sample Output

```

13:13:44          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate ADACMP Definitions -          Page: 1
Input file: GEN-EXAM-ADACMP-INPUT
File type : Sequential file

  Cnt Ty L          Field name          F Length D U DB S Occ Adabas
                    Corresponding Adabas Field          F Leng.
-----
1  1 ELE-N-9V5          N    7.05          N    U    12
  2 ELE-N-9V5          N    9.05          AB N    U    14
  >>> FIELD LENGTH DEFINITION IS DIFFERENT <<<
  >>> ADABAS FIELD LENGTH IS DIFFERENT <<<
3  1 ELE-PS-5V2         PS   5.00          N    P    3
  3 ELE-PS-5V2         PS   5.02          AE N    P    4
  >>> FIELD LENGTH DEFINITION IS DIFFERENT <<<
  >>> ADABAS FIELD LENGTH IS DIFFERENT <<<
8  1 SB-ELE-A-5         A    5.00          A    A    5
  SB 1 SB-ELE-A-5      A    5.00          AJ N    A    5
  >>> FIELD TYPE IS DIFFERENT <<<
9  PE 1 PC-OCC-7                2
  PC 1 PC-OCC-7                AL    7
  >>> NUMBER OF OCCURRENCES IS DIFFERENT <<<

```

```

13:18:34          ***** P R E D I C T 4.3.1 *****          2003-05-31
                  - Generate ADACMP Definitions -                Page:    2

File ID .. GENERATION-EXAMPLE

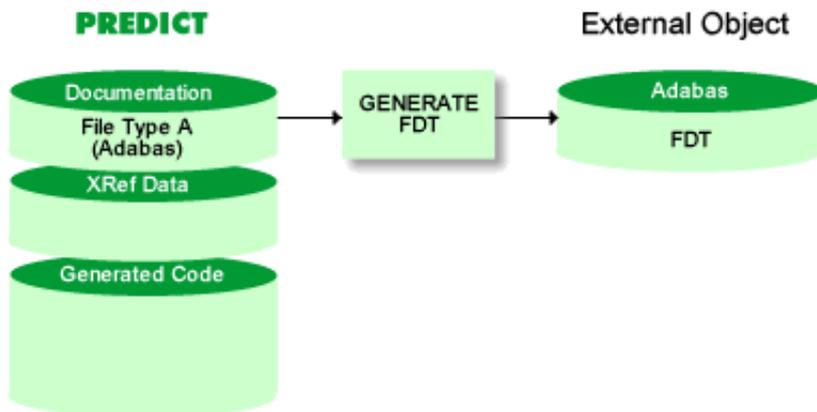
ADACMP COMPRESS
ADACMP FILE=171
ADACMP FORMAT='AB,12,AD,AE,3,AF1-5,AG,AH,AI,AJ,AM1,AN1(1-11),AP1,AQ1'
ADACMP FORMAT=' ,AM2,AN2(1-11),AP2,AQ2,AS,AU.'
ADACMP LRECL=242
ADACMP NUMREC=0
ADACMP MINISN=1
ADACMP DEVICE=3390
ADACMP FNDEF='01,AA'                GROUP-1                *****
ADACMP FNDEF='02,AB,14,U,NU'        ELE-N-9V5              2003-05-31
ADACMP FNDEF='02,AC'                GR-IN-GROUP            13:18:34
ADACMP FNDEF='03,AD,4,B,DE,FI'      ELE-B-4                *****
ADACMP FNDEF='03,AE,4,P,NU'        ELE-PS-5V2
ADACMP FNDEF='03,AF,4,B,NU,MU'      MU-B-4
ADACMP FNDEF='02,AG,42,A,NU'        ELE-A-42
ADACMP FNDEF='02,AH,8,G,FI'        ELE-F-8
ADACMP FNDEF='02,AI,3,B,NU'        ELE-B-3
ADACMP SUBFN='AJ=AG(10,14)'        SB-ELE-A-5
ADACMP SUPDE='S1=AI(1,3),AG(20,36),-' SP-DE-A-24
ADACMP      'AE(1,4)'
ADACMP FNDEF='01,AL,PE'              PC-OCC-7
ADACMP FNDEF='02,AM,10,U,DE,NU'     PC-ELE-DE-NS-7V3
ADACMP FNDEF='02,AN,4,P,DE,NU,MU'   PC-MC-PS-6V1
ADACMP FNDEF='02,AO'                PC-GR
ADACMP FNDEF='03,AP,2,B,FI'         PC-ELE-I-2
>>> CHANGE: FORMAT NOT SUPPORTED BY GIVEN ADABAS VERSION. <<<
ADACMP FNDEF='03,AQ,14,P,NU'        PC-PS-20V7
ADACMP FNDEF='02,AR,4,G,FI'        PC-ELE-F-4
>>> WARNING: FIELD NOT FILLED BY INPUT FILE. <<<
ADACMP FNDEF='01,AS,4,P,NU'        ELE-D
>>> CHANGE: FORMAT 'D'/'T' CHANGED TO 'P'. <<<
ADACMP FNDEF='01,AT,7,P,NU'        ELE-T
>>> WARNING: FIELD NOT FILLED BY INPUT FILE. <<<
>>> CHANGE: FORMAT 'D'/'T' CHANGED TO 'P'. <<<
ADACMP FNDEF='01,AU,1,B,NU'        ELE-L
ADACMP HYPDE='04,S2,4,F,FI,MU,PE=AP,AF' HQ-DE-I-4
>>> CHANGE: OPTION FI REQUIRED BY ADABAS. <<<

```

```
//  
ADALOD LOAD  
ADALOD FILE=171  
ADALOD NAME=GENERATION-EXAMP  
ADALOD ASSOPFAC=10  
ADALOD DATAFRM=YES  
ADALOD DATAPFAC=10  
ADALOD DSDEV=3390  
ADALOD DSREUSE=YES  
ADALOD DSSIZE=233B  
ADALOD ISNREUSE=NO  
ADALOD MAXISN=3391  
ADALOD MINISN=1  
ADALOD NISIZE=80B  
ADALOD NUMREC=0  
ADALOD SORTDEV=3380  
ADALOD SORTSIZE=25  
ADALOD TEMPDEV=3380  
ADALOD TEMPsize=30  
ADALOD UISIZE=41B  
  
DIC1800 SUMMARY:      27 FIELD(S) PROCESSED  
DIC1825 WARNING:      2 WARNING(S) IN GENERATION FUNCTION ISSUED  
DIC1815 WARNING:      4 CHANGE(S) IN FIELD DEFINITIONS MADE
```

Adabas File

The function generates an Adabas FDT from a Predict file object of type A (Adabas).

**Note:**

If Natural Security and Adabas Online Services are installed and function protection for Predict is activated, permission is required to generate an Adabas file.

Note for UNIX, Windows NT and VMS Users:

With this function you can generate input for the ADAFDU utility to create a new file in your Adabas database. For existing files the FDT and the documentation of the file are compared and the input, necessary to adapt the existing FDT for the utilities ADAINV and ADADBM, is generated. The utilities are started immediately. This method automatically creates a connection between external and documentation object. Before you can delete the documentation object, you must either delete the external object (with the administration function Purge) or disconnect the external and documentation object (with function Disconnect). See also section Handling of External and documentation Objects in this documentation.

A file generated with this method is handled like a file created with Adabas Online Services.

This section covers the following topics:

- Calling the Function
- Generate Adabas File in Batch Mode
- Sample Output
- Sample Output - Continued

Calling the Function

The Generate Adabas file screen is displayed with function G and code AF in a Predict main menu, or with the command GENERATE FDT.

On mainframes, the following screen is displayed:

```

13:21:46          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Generate Adabas file -

Current VM ..... HOME

File ID .....*
Phys. file number .....
Contained in DA .....
Phys. database number ..... 36

Load file into database ... Y (Y/N)   Replace if only FDT exists .... N (Y/N)
Update/add fields (FDT) ... N (Y/N)
Update file parameters ... N (Y/N)   Replace if no records loaded .. N (Y/N)

Change file number ..... N (Y/N)
Format DS/Index ..... Y (Y/N)
Adabas password .....

List actions ..... Y (Y/N)

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJl Flip Print Impl AdmFi SelFi Prof Main
    
```

On Windows and UNIX platforms, the following screen is displayed:

```

13:28:33          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Generate Adabas file -

Current VM ..... HOME

File ID .....*
Phys. file number .....
Contained in DA .....
Phys. database number .....

Load file into database ... Y (Y/N)   Replace if only FDT exists .... N (Y/N)
Update/add fields (FDT) ... N (Y/N)
Update file parameters ... N (Y/N)   Replace if no records loaded .. N (Y/N)

Change file number ..... N (Y/N)   Adabas version .....* U1
Format DS/Index ..... Y (Y/N)   Contiguous AC ..... N (Y/N)
Adabas password .....          Contiguous DS ..... N (Y/N)
                                   Contiguous NI ..... N (Y/N)
List actions ..... Y (Y/N)   Contiguous UI ..... N (Y/N)

Command ===>
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

The following parameters can only be set with the function Defaults > Generation Defaults for object type AF.

Default Parameters

Delay after stop users	<p>Enter a value of up to 99 seconds. This is the time the system waits before actually performing the generation if parameter Stop users using file is set to Y. See below.</p> <p>Due to Adabas load or too much network traffic, it is possible that not all users have been stopped before the function Generate Adabas file is executed. This delay makes certain that all users really have been stopped before an Adabas file is generated.</p>
Execute online invert	<p>Y If set to N, no online invert function is executed.</p> <p>Note: This parameter applies only when generating for Adabas databases of version 7 or above.</p>
Only if TOPISN less than	<p>This function is only available if Execute online invert is set to Y. You can specify a value for TOPISN. The default value is 9999999999.</p> <p>Note: This parameter applies only when generating for Adabas databases of version 7 or above.</p>
Delay after online invert	<p>The online invert function of Adabas version 7 is executed as an asynchronous task. Only one online invert task may be active at a given time.</p> <p>Enter a value of up to 99 seconds. This is the time the system waits after one online invert before performing the next.</p> <p>Note: This parameter applies only when generating for Adabas databases of version 7 or above.</p>
Stop users using file	<p>Y All current use of the Adabas file is stopped before a generation run is executed. This parameter must be set to Y on a separate terminal if a file currently used by other users is to be regenerated.</p> <p>N The function is terminated if other users are accessing the file.</p> <p>Note: If Natural Security and Adabas Online Services are installed and function protection for Predict is activated, permission is required to stop users using an Adabas file.</p>
Current VM	<p>Adabas files can only be generated from Predict file objects contained in databases accessible from the current virtual machine.</p>
Parameters	
File ID, Phys. file number, Contained in DA, Phys. database number	<p>The Adabas file definitions can be generated for one specific Adabas file in a database. This file can be identified by the fields File ID, Phys. file number, Database ID or Phys. database number. If parameters do not uniquely identify a file, a selection screen appears.</p>
Load file into database	<p>Y If no file already exists with given file number, a new file is created in the Adabas database.</p>

Update/add fields (FDT)	Y Fields documented in Predict and missing in the FDT are added to the Adabas file. The field length in the FDT is adapted to the value in the Predict object, and fields defined as descriptors in the FDT but not in the Predict documentation are released from descriptor status.
Update file parameters	Y File parameters, including the file space allocation, are adapted to the documentation. If the Adabas file is physically coupled to other files and these relationships are not documented, the files are uncoupled.
Change file number	Y If the Predict file object was renumbered, the Adabas file is renumbered.
Format DS/Index	DATA storage is to be physically formatted.
Adabas password	Must be provided if the Adabas file (or only an FDT) already exists in the Adabas database and is protected by an Adabas security definition.
List actions	Y Actions which are to be performed to adapt the Adabas file to the Predict specification are listed. Some of the actions may not yet be performed by this generation function: they must be run as batch utilities. The list shows all parameters which are used for generation. Parameters which are invalid or currently not able to be implemented are marked accordingly.
Replace if only FDT exists	Y If a file with this file number once existed in the database and was deleted with the KEEPFDT option, the remaining FDT is overwritten. Note: The FDT of the file will not be deleted if the generation terminates abnormally, for example if the field list in Predict contains invalid field definitions.
Replace if no records loaded	Y The Adabas file parameters are replaced, because it is not possible to modify all parameters of an existing file. Replacement is only possible if the Adabas file is empty and is not physically coupled to other Adabas files. The current FDT is kept.
Additional Parameters for UNIX	
Adabas version	The version of Adabas for which the external object is to be generated. Enter an asterisk for valid values or see Adabas Version for more information.
Contiguous AC, DS, NI, UI	Setting of space allocations in Adabas utility ADAFDU for Address Converter, Data Storage, Normal Index and Upper Index. See your Adabas Utilities documentation for more information.

Generate Adabas File in Batch Mode

Command: GENERATE FDT

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Phys. file number	FNR	2
Database ID	DB	3
Phys. database number	DBNR	4
Load file into database	LOAD	5
Replace if only FDT exists	REPLACE-FDT	6
Update/add fields (FDT)	UPD-FDT	7
Update file parameters	UPD-PARM	8
Replace if no records loaded	REPLACE-EMPTY	9
Note: Value exists for compatibilty reasons only	REPLACE-CRIT	10
Change file number	RENUMBER	11
Format DATA storage	DATAFRM	12
Adabas password	PSW	13
List actions	LIST	14
Stop users using file	STOPF	15

Sample Output

Action	Parameter	Remark
	13:07:37	***** P R E D I C T 4.3.1 ***** - Generate Adabas file -
	File ID GENERATION-EXAMPLE	PFnr ... 171
	Database ID .. DEMO-DB	PDBnr .. 180
DEFFDT	FILE=171	
	FNDEF='01,AA'	
	Field: GROUP-1	
	FNDEF='02,AB,14,U,NU'	
	Field: ELE-N-9V5	
	FNDEF='02,AC'	
	Field: GR-IN-GROUP	
	FNDEF='03,AD,4,B,DE,FI'	
	Field: ELE-B-4	
	FNDEF='03,AE,4,P,NU'	
	Field: ELE-PS-5V2	
	FNDEF='03,AF,4,B,NU,MU'	
	Field: MU-B-4	
	FNDEF='02,AG,42,A,NU'	
	Field: ELE-A-42	
	FNDEF='02,AH,8,G,FI'	
	Field: ELE-F-8	
	FNDEF='02,AI,3,B,NU'	
	Field: ELE-B-3	
	FNDEF='01,AL,PE'	
	Field: PC-OCC-7	
	FNDEF='02,AM,10,U,DE,NU'	
	Field: PC-ELE-DE-NS-7V3	
	FNDEF='02,AN,4,P,DE,NU,MU'	
	Field: PC-MC-PS-6V1	
	FNDEF='02,AO'	
	Field: PC-GR	
	FNDEF='03,AP,2,B,FI'	
	Field: PC-ELE-I-2	
	FNDEF='03,AQ,14,P,NU'	
	Field: PC-PS-20V7	
	FNDEF='02,AR,4,G,FI'	
	Field: PC-ELE-F-4	
	FNDEF='01,AS,4,P,NU'	
	Field: ELE-D	
	FNDEF='01,AT,7,P,NU'	
	Field: ELE-T	
	FNDEF='01,AU,1,B,NU'	
	Field: ELE-L	
	SUBFN='AJ=AG(10,14)'	
	Field: SB-ELE-A-5	
	SUPDE='S1=AI(1,3),AG(20,36),-'	
	'AE(1,4)'	
	Field: SP-DE-A-24	
	HYPDE='04,S2,4,F,MU,PE=AP,AF'	
	Field: HQ-DE-I-4	

Sample Output - Continued

```
13:07:37          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate Adabas file -

File ID ..... GENERATION-EXAMPLE          PFnr ... 171
Database ID .. DEMO-DB                    PDBnr .. 180

Action   Parameter                          Remark

LOADFILE NOACEXTENSION=NO
          MIXDSDEV=NO
          DATAFRM=YES
          MAXISN=3391
          NAME=GENERATION-EXAMPLE
          MINISN=1
          ASSOPFAC=10
          DATAPFAC=10
          PGMREFRESH=NO
          ISNREUSE=NO
LOADFILE DSREUSE=YES
          MAXDS=0
          MAXNI=0
          MAXUI=0
          CIPHER=NO
          MAXRECL=5060
          UISIZE=41B
          NISIZE=80B
          DSSIZE=233B
          DSDEV=3390
```

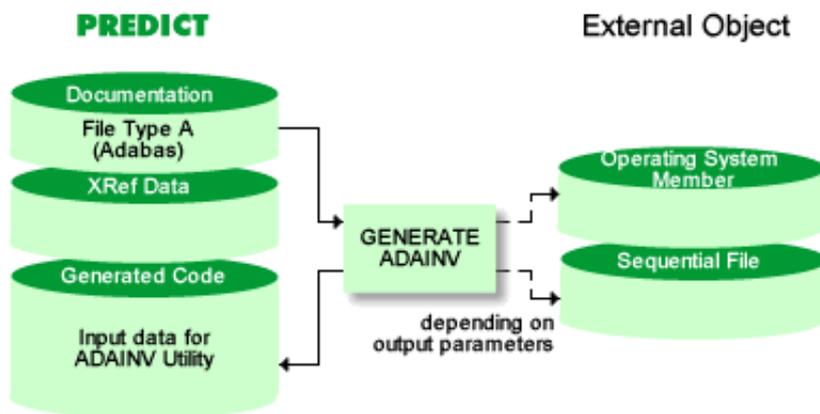
ADAINV Definitions

The function Generate ADAINV Definitions generates input data for the Adabas utility ADAINV from Predict file objects of type A.

The function applies to the following Adabas versions:

- IBM/Siemens Version 5.1 or above
- UNIX Version 2.1 or above
- VMS Version 2.1 or above

Predict checks against the implemented Adabas file which descriptors are already defined and generates the ADAINV definition only for the additional descriptors in Predict.



If Adabas Online Services are installed, ADAINV couple cards are generated additionally if file relations of type C (physically coupled) are documented but not implemented for the file.

When using Adabas version 7 or above on mainframes with Adabas Online Services installed or Adabas on a Windows or UNIX platform, the invert function can be executed as part of the Generate Adabas file function. In this case it is not necessary to adapt the FDT in a three step process (i.e. generate Adabas file, generate ADAINV and execute ADAINV).

In all other configurations, before using the ADAINV utility with Predict, it is recommended to first generate the Adabas FDT, then the ADAINV definitions.

The output generated from this function is used as input for the ADAINV functions INVERT and COUPLE in one member. The output must be punched and edited as necessary. See Sample Output.

This section contains:

- Calling the Function
- Generate ADAINV Definitions in Batch Mode
- Sample Output

Calling the Function

The Generate ADAINV Definitions screen is displayed with function code G and object code AI in a Predict main menu, or with the command GENERATE ADAINV.

```

09:13:47          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Generate ADAINV Definitions -

Current VM ..... HOME
File ID .....*
Phys. File number .....
Contained in DA .....
Phys. Database number ....

Save as member .....          Save in library .... INVLIB
Overwrite option ..... Y      (Y,N)      Op. system member ..
Punch / output .....* N

List generated code ..... Y      (Y,N)      Adabas version ....* I7
SORTSIZE .....          C/B ... C      SORTDEV .....*
TEMPSIZE .....          C/B ... C      TEMPDEV .....*
Work pool size .....          KB/B .. KB      Cipher code .....
Prefetch buffer size .....          KB/B .. KB      Adabas password ....
Handle UQ conflict .....* A

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJl Flip Print Impl AdmFi Selfi Prof Main

```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

The parameter below can be changed in the Modify ADAINV Defaults screen. See Generation Defaults.

Presetting	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated. See Parameters Specifying the Form of Output for more information.
Parameters	
The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output, Workfile name and List generated code are described in the section Parameters Specifying the Form of Output.	
Current VM	Read-only field. ADAINV definitions can only be generated from Predict file objects contained in databases accessible from the current virtual machine.
File ID, Phys. file number, Contained in DA, Phys. database number	ADAINV definitions are generated for an Adabas file. The file must not be marked as Adabas system file. The file is identified either by file ID or physical file number in the database. The database can be identified either by database ID or physical database number. If one physical file number and one physical database number are not identified uniquely, a selection screen appears. The identified Adabas file must be implemented, because Predict performs the Adabas LF commands to check which descriptors are already defined.
SORTSIZE, TEMPSIZE	Both values must be entered (cylinders or blocks).
SORTDEV, TEMPDEV	Only required if the value is different from that specified with the ADARUN DEVICE parameter.
Work pool size	Both values can be specified in bytes or in kilobytes.
Prefetch buffer size	If no value is specified, the default size is taken.
Cipher code	Applies only to files marked in Predict as ciphered. Specifies the value of the cipher code parameter to be generated in the ADAINV control statement.
Adabas password	If the file specified is security protected, an appropriate Adabas password must be supplied.
Handle UQ conflict	Determines system response if descriptor is not unique. This parameter only applies to Adabas versions VMS 2.1 or above and UNIX 1.2 or above. A Abort R Reset

Generate ADAINV Definitions in Batch Mode

Command: GENERATE ADAINV

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Phys. file number	FNR	2
Database ID	DB	3
Phys. database number	DBNR	4
Save as member	MEM	5
Save in library	LIB	6
Overwrite option	REPLACE	7
Op. system member	OS-MEMBER	8
Punch / output	PUNCH	9
List generated code	LIST	10
Adabas version	ADA-VER	11
SORTSIZE	SORTSIZE	12
C/B	SORTSIZE-TYPE	13
SORTDEV	SORTDEV	14
TEMPSIZE	TEMPSIZE	15
C/B	TEMPSIZE-TYPE	16
TEMPDEV	TEMPDEV	17
Work pool size	LWP	18
KB/B	LWP-TYPE	19
Cipher code	CIPHER	20
Prefetch buffer size	LPB	21
KB/B	LPB-TYPE	22
Adabas password	PSW	23
Handle UQ conflict	UQ-CONFLICT	24
Workfile name (see note below)	WORKFILE-NAME	25
If Entire System Server is used		
- DB-ID	NP-DBID	26
- Dataset	NP-DSNAME	27
- Volume	NP-VOLSER	28
- Library	NP-LIB	29
- Sublibrary	NP-SUBLIB	30
- Member type	NP-MEMTYPE	31
- VSAM catalog	NP-VSAMCAT	32

Note:

You can not specify a value other than blank (' ') for WORKFILE-NAME on mainframes.

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

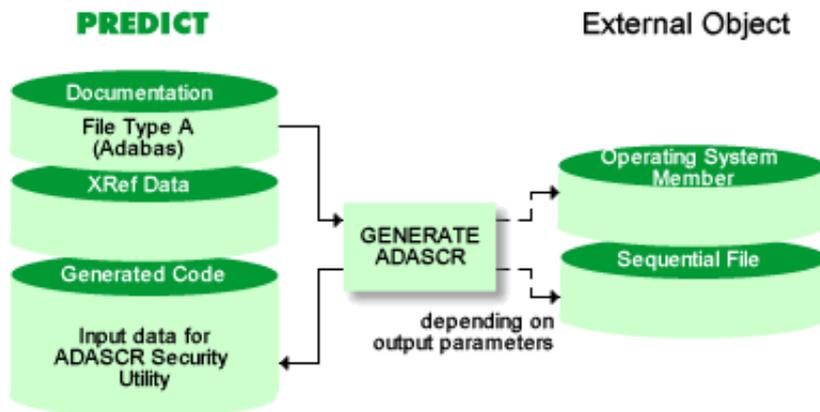
Sample Output

```
13:50:56          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate ADAINV Definitions -                    Page: 1
ADAINV INVERT FILE=171
ADAINV TEMPSIZE=60
ADAINV SORTSIZE=40,SORTDEV=3380
ADAINV FIELD='AD'
ADAINV HYPDE='04,S2,4,F,FI,MU,PE=AP,AF'
/*
ADAINV COUPLE
ADAINV TEMPSIZE=60
ADAINV SORTSIZE=40,SORTDEV=3380
ADAINV FILES=171,172,DESCRIPTOR='AD,AA'
```

```
DIC1800 SUMMARY:      27 FIELD(S) PROCESSED
```

ADASCR Definitions

This function generates definitions for the ADASCR security utility from Predict file objects of type A.



This section contains:

- Calling the Function
- Generate ADASCR Definitions in Batch Mode
- Sample Output

Calling the Function

The Generate ADASCR Definitions screen is displayed with function code G and object code AS in a Predict main menu, or with the command GENERATE ADASCR.

```

09:19:13          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Generate ADASCR Definitions -

File ID .....*
Phys. File number .....
Contained in DA .....
Phys. Database number ..

Save as member .....          Save in library .... SCRLIB
Overwrite option ..... Y    (Y,N)          Op. system member ..
Punch / output .....* N

List generated code .... Y    (Y,N)          Adabas version ....* I1

Code .....

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJ1 Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

The parameter below can be changed in the Modify ADASCR Defaults screen. See Generation Defaults.

Presetting	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated. See Parameters Specifying the Form of Output for more information.
Parameters	
The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output, Workfile name and List generated code are described in Parameters Specifying the Form of Output.	
File ID, Phys. file number, Contained in DA, Phys. database number	The Adabas Security definitions can be generated for one specific Adabas file in a database. This file is uniquely defined by the physical file number and the physical database number. If the parameters entered are not unique, a selection screen appears. It is also possible to generate the definitions for an Adabas file which is not linked to any database. In this case the currently defined file attributes are taken.
Adabas version	The version of Adabas for which the external object is to be generated. Enter an asterisk for valid values or see Adabas Version for more information.
Code	If a code is entered, the parameter CODE=code is generated.

Generate ADASCR Definitions in Batch Mode

Command: GENERATE ADASCR

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other taken.

Field	Keyword	Position
File ID	FILE-ID	1
Save as member	MEM	2
Save in library	LIB	3
Overwrite option	REPLACE	4
Op. system member	OS-MEMBER	5
List generated code	LIST	6
Punch / output	PUNCH	7
Adabas version	ADA-VER	8
Code	PSW	9
phys. File-Nr.	FNR	10
Database ID	DB	11
phys. Database-Nr.	DBNR	12
Workfile name (see note below)	WORKFILE-NAME	13
If Entire System Server is used		
- DB-ID	NP-DBID	14
- Dataset	NP-DSNAME	15
- Volume	NP-VOLSER	16
- Library	NP-LIB	17
- Sublibrary	NP-SUBLIB	18
- Member type	NP-MEMTYPE	19
- VSAM catalog	NP-VSAMCAT	20

Note:

You can not specify a value other than blank (' ') for WORKFILE-NAME on mainframes.

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

Sample Output

```
13:25:14          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate ADASCR Definitions -                Page: 1

File ID .. GENERATION-EXAMPLE

ADASCR PROTECT FILE=171,ACC=2,UPD=3
ADASCR PROTECT FILE=171,NAME=AB,ACC=4,UPD=15
ADASCR PROTECT FILE=171,NAME=AD,ACC=6,UPD=12
ADASCR PROTECT FILE=171,NAME=AE,ACC=3,UPD=13
ADASCR PROTECT FILE=171,NAME=AF,ACC=3,UPD=12
ADASCR PROTECT FILE=171,NAME=AP,ACC=5,UPD=7
ADASCR PROTECT FILE=171,NAME=AR,ACC=2,UPD=14
  >>> File access level inserted.                                <<<
  for field: PC-ELE-F-4

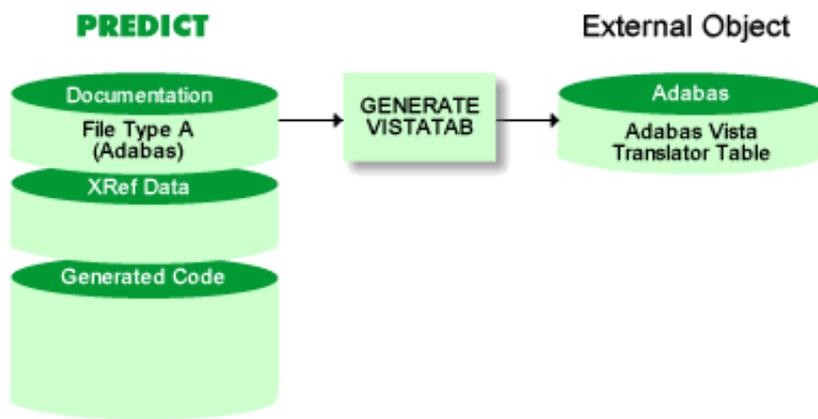
DIC1800 SUMMARY:      27 FIELD(S) PROCESSED
```

Adabas Vista Translation Table

The function implements Vista elements of Predict file objects of type A (Adabas) in the translation tables of Adabas Vista.

The Generate Vista table function has to be executed for each Adabas file that is to be implemented.

The Generate Vista table function can either be executed for a specific Adabas file or for all Adabas files located in databases in a virtual machine belonging to the current network. The translation elements on database level are also generated.

**Note:**

See the section Adabas Vista in the Predict and Other Systems documentation for a detailed description of using Predict for distributed data storage.

This section contains:

- Calling the Function
- Generate Vista Table in Batch Mode

Calling the Function

The Generate Vista table screen is displayed with function G and code AT in a Predict main menu, or with the command GENERATE VISTATAB.

The actions necessary to implement the Vista elements can be listed by setting the parameter List action to Y. Additional confirmation is requested before the Vista elements are actually implemented.

```

13:32:12          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Generate Vista table -

Current VM ..... HOME

File ID .....*

Add Vista elements ..... Y (Y/N)
Update Vista elements ..... Y (Y/N)

Check FDT ..... N (Y,N)
Adabas password .....

Purge all Vista elements not documented .. N (Y/N)

List actions ..... Y (Y/N)
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Parameters	
Current VM	Determines which Vista elements are processed. A read only field. The current virtual machine is set in the Maintenance options of the Modify User Defaults function. See the appropriate document of section General Information in the Introduction to Predict documentation .
File ID	File to be processed. If file ID is specified, all Vista elements of this file and all Vista elements of databases in the current virtual machine the file is contained in, are processed. If no file ID is specified, all Vista elements of all files and all Vista elements of all databases of the current virtual machine, are processed.
Add Vista elements	Y Entries for Vista elements that are not yet contained in the translation table are generated.
Update Vista elements	Y Existing entries for Vista elements are updated if different.
Check FDT	Y Checks if the physical files that form the logical partitioned file all have the same FDT.
Purge all Vista elements not documented	Entries in the translator table without corresponding Vista elements in Predict file objects are purged.
Adabas password	Must be provided if Check FDT is set to Y or if an Adabas file is protected by an Adabas security definition.
List actions	Y Actions which are to be performed to (re)generate an existing Vista table.

Generate Vista Table in Batch Mode

Command: GENERATE VISTATAB

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Add Vista elements	ADD-STARTAB	2
Update Vista elements	UPD-STARTAB	3
Check FDT	CHECK-FDT	4
Adabas password	PSW	5
Purge all Vista elements not documented	PUR-STARTAB	6
List actions	LIST	7

This function is not yet implemented.

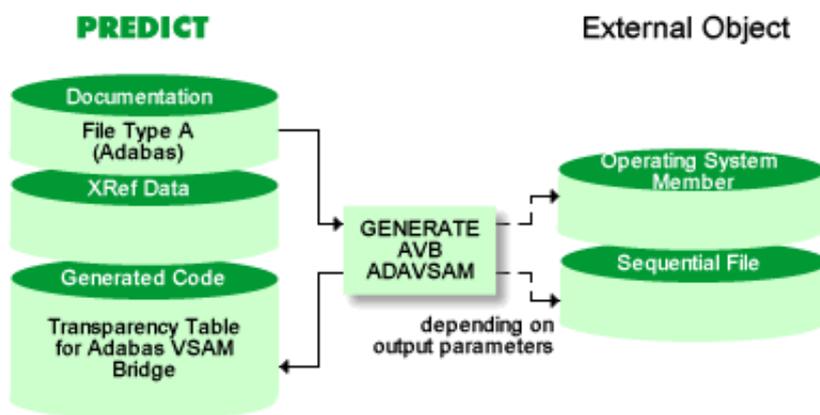
Adabas VSAM Bridge Transparency Table

The Adabas VSAM Bridge supports conversion from VSAM files of type KSDS (key sequenced data set) organization to Adabas files without the need to change existing COBOL programs. Both single-record and multi-record files are supported. The conversion of the VSAM record structures into an Adabas file is described by a transparency table which can be generated with Predict.

For details of the application of Adabas VSAM Bridge transparency tables, see the **Adabas VSAM Bridge Installation and Operations documentation**.

Note:

The optional parameters KEYB, PCKKEY, PFOFF1, PKOFF2, PREFSZE and RESET are not supported.



This section contains:

- Prerequisites
- Rules Applying to the Design of the Adabas Files
- Calling the Function
- Generate Transparency Table in Batch Mode
- Examples
- Sample File Definition
- Sample Output

Prerequisites

Before Predict can be used to generate a transparency table for a VSAM file and an Adabas file, both files must be defined in Predict file and field objects.

- A single-record VSAM file must be completely defined in a file of type V and corresponding fields.
- A multi-record VSAM file must have a file of type V with field objects at least for the primary key, alternate keys and indicator field. The records are identified by the indicator field: its field short name must always be II.

The different record structures must be defined as separate logical VSAM files in separate files of type L and respective fields.

The indicator value of the records must be entered as V-PREFIX to the logical VSAM file objects.
 The primary key, alternate keys and indicator field must be defined with the same format and length in all files.
 The offset must be the same in all records.

A VSAM file, single-record or multi-record, is always converted into one Adabas file. Corresponding fields in an Adabas file, a VSAM file and logical VSAM files are connected using the field short name.

Rules Applying to the Design of the Adabas Files

The following points should be considered when designing an Adabas file:

- The format buffer definition in the transparency table is generated only for Adabas fields on level 1 so MU-fields must always be defined on level 1;
- The group structure in the VSAM file(s) and the Adabas file are independent but it must be possible to generate a format buffer for all VSAM record structures from the Adabas fields or groups on level 1;
- PE groups with more than 99 occurrences and MU fields with more than 191 occurrences must be split up in the Adabas file into several contiguous segments. The first segment must have the same field short name as that PE group or MU field has in the VSAM file; later segments must have field short names that do not exist in the VSAM file. All segments except the last must have 64 occurrences.
- The maximum length of the indicator field is 20 bytes and the indicator values must not contain blanks.

Calling the Function

The Generate Transparency Table screen is displayed with function code G and object code AV in a Predict main menu, or with the command GENERATE ADAVSAM.

```

09:28:23          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Generate Transparency Table -

VSAM file ID .....*

Related Adabas file ID ...

Save as member .....          Save in library ..... AVBLIB
Overwrite option ..... Y (Y,N)      Op. system member ....
Punch / output .....* N          Save incorrect code .. N (Y,N)

List generated code ..... Y (Y,N)      Adabas version .....* I1
List key table ..... Y (Y,N)
List Adabas relations .... Y (Y,N)
List VSAM relations ..... Y (Y,N)
Change Adabas file ..... N (Y,N)

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJel Flip Print Impl AdmFi SelFi Prof Main
  
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

The parameter below can be changed in the Modify Adabas/VSAM Bridge Defaults screen. See Generation Defaults.

Presetting	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated.
Parameters	
The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output and List generated code are described in Parameters Specifying the Form of Output.	
VSAM file ID	ID of the Predict VSAM file object from which the definitions are to be generated. The file must be of type V (VSAM).
Related Adabas file ID	ID of the Predict file object from which the definitions are to be generated. The file must be of type A (Adabas). This parameter can be omitted if generation of a transparency table for the same two files has been started before.
Save incorrect code	Y A punch or save is performed even if errors are found.
Adabas version	Version of Adabas for which the transparency table is to be generated. Enter an asterisk for valid values or see Adabas Version for more information.
List key table	Y Summary of all VSAM keys is to be listed. In the case of a multi-record type file, the definitions of the indicator and the logical VSAM file with the indicator value are listed.
List Adabas relations	Y Attributes of the fields of the Adabas file and their corresponding VSAM fields are to be listed.
List VSAM relations	Y The fields of the different VSAM record types and corresponding VSAM fields are to be listed.
Change Adabas file	Y A transparency table is generated even if one has already been generated for the same VSAM file and a different Adabas file, provided that the newly specified Adabas file is not related to any other VSAM file.

Generate Transparency Table in Batch Mode

Command: GENERATE ADAVSAM.

Enter parameters on next line in positional or keyword form. VSAM File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
VSAM file ID	VSAM-FILE-ID	1
Related Adabas file ID	Adabas-FILE-ID	2
Save as member	MEM	3
Save in library	LIB	4
Overwrite option	REP-OPT	5
Op. system member	OS-MEMBER	6
List key table	LIST-KEY	7
List Adabas relations	LIST-Adabas	8
List VSAM relations	LIST-VSAM	9
List generated code	LIST-TT	10
Change Adabas file	REPLACE-ADA	11
Punch / output	PUNCH	12
Save incorrect code	SAVE-CODE	13
Adabas version	ADA-VER	14
If Entire System Server is used		
- DB-ID	NP-DBID	15
- Dataset	NP-DSNAME	16
- Volume	NP-VOLSER	17
- Library	NP-LIB	18
- Sublibrary	NP-SUBLIB	19
- Member type	NP-MEMTYPE	20
- VSAM catalog	NP-VSAMCAT	21

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

Examples

Layout lists of the four files we have used as examples (one Adabas file, one physical VSAM file and two logical VSAM files) are given below. They are followed by part of the output produced when generating a transparency table for these files: the key table; the beginning of the description of the Adabas file; the beginning of the description of one logical VSAM file; and, finally, the transparency table. In the description of the logical VSAM file, the column headed FB DB gives the field short name used for the format buffer and the column headed OFF FB gives the offset of the field in the group used for the format buffer.

Sample File Definition

This sample file definition was used to create the sample output on.

```

13:26:11          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - List File with Children -

File ID ..... AZ-QA-AVB-A-ART
-----
ADABAS test file ARTICLE
for ADABAS VSAM BRIDGE (AVB)

  Cnt  Ty L Field ID          F      Length  Occ D DB S      Off  Off
                                     Dec  Hex

  **    *** FIX PART OF VSAM RECORDS ***

  1 GR 1 ART-FIX                AA        0      0
  2   2 ART-DESC                A      32.0    D AB N      0      0
  3   2 DATE-ADD                N       6.0    AC N     32     20
  4   2 DATE-MODIFIED          N       6.0    AD N     38     26
  5 GR 2 ART-NR                AE        44     2C
  6   3 ART-REC-KEY            A       5.0    II N     44     2C
  7   3 ART-GROUP              A       4.0    D AF N     49     31
  8   3 DEPARTMENT             N       3.0    AG      53     35
  9   3 SEQ-NR                 N       2.0    AH N     56     38
 10   3 CHECKDIGIT             N       2.0    AI N     58     3A
 11   2 SPECIALIST             A      30.0    D AJ N     60     3C
 12   2 REC-TYPE               N       2.0    D MI N     90     5A
 13 SP 1 ART-KEY                A      16.0    D SK N

      >>> FIELD(S) SKIPPED FOR RECORDBUFFER STRUCTURE <<<

  **    *** SALES INFORMATION ***

 17   1 PRICE                   N       5.2    AM N     92     5C
 18 MU 1 DISCOUNT             N       5.2     6 AN N     99     63
 19   1 STOCKS-NR-DELIVERY-CNT N       2.0    B1 N    141     8D
 20 MU 1 STOCKS-NR-DELIVERY    A      12.0   20 D AO N    143     8F
 21 PE 1 STATISTICS-MONTH      N       5.0   12 AP     383    17F
 22   2 TURNOVER               N       5.0    AQ N     383    17F
 23   2 QUANTITY               N       5.0    AR N     388    184

  **    *** COMPONENTS OF ARTICLE ***

 24   1 COMPONENTS-NR          N       2.0    AS N    503    1F7
 25 PE 1 COMPONENTS            A       64    AT     505    1F9
 26   2 COMPONENT-ART-NR       A      12.0    AU N    505    1F9
 27   2 COMPONENT-QUANTITY     P       5.0    AV     517    205
 28 PE 1 COMPONENTS-C          N       35    AW    1465    5B9
 29   2 COMPONENT-ART-NR-C     A      12.0    AX N    1465    5B9
 30   2 COMPONENT-QUANTITY-C   P       5.0    AY N    1477    5C5
 31   1 STOCKS-CNT            N       2.0    B2 N    1990    7C6
 32 PE 1 STOCKS                A       30    AZ     1992    7C8
 33   2 STOCK-NR              A      12.0    A0 N    1992    7C8
 34   2 STOCK-QUANTITY         I       4.0    A1 F    2004    7D4

      *** TOTAL LENGTH ***                      2472   9A8

*** End of report ***

```

```

13:41:46          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - List File with Children -

File ID ..... AZ-QA-AVB-V-ART
Type ..... VSAM file
-----

Abstract
  VSAM test file article
  FOR ADABAS VSAM BRIDGE (AVB)

  Cnt  Ty L Field ID          F      Length  Occ D DB S      Off  Off
                                Dec  Hex

  1 GR 1 ART-FIX                AA          0    0
  2   2 ART-DESC                A      32.0    A AB N          0    0
  3   2 DATE-ADD                 N       6.0    AC N          32   20
  4   2 DATE-MODIFIED            N       6.0    AD N          38   26
  5 GR 2 ART-NR                  AE          44   2C
  6   3 ART-REC-KEY              A       5.0    II N          44   2C
  7   3 ART-GROUP                A       4.0    AF N          49   31
  8   3 DEPARTMENT               N       3.0    AG           53   35
  9   3 SEQ-NR                    N       2.0    AH N          56   38
 10   3 CHECKDIGIT               N       2.0    AI N          58   3A
 11   2 SPECIALIST                A      30.0    A AJ N          60   3C
 12   2 REC-TYPE                  N       2.0    MI N          90   5A
 13 SP 1 REC-ART-KEY              A      16.0    P SK          44   2C

          *** TOTAL LENGTH ***                      92   5C

*** End of report ***
    
```

```

13:43:40          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - List File with Children -

File ID ..... AZ-QA-AVB-L-ART-COMP
Type ..... Logical VSAM
-----

```

Cnt	Ty	L	Field ID	F	Length	Occ	D	DB	S	Off Dec	Off Hex
** *** FIXED PART ***											
1	GR	1	ART-FIX					AA		0	0
2		2	ART-DESC	A	32.0			A	AB N	0	0
3		2	DATE-ADD	N	6.0				AC N	32	20
4		2	DATE-MODIFIED	N	6.0				AD N	38	26
5	GR	2	ART-NR						AE	44	2C
6		3	ART-REC-KEY	A	5.0				II N	44	2C
7		3	ART-GROUP	A	4.0				AF N	49	31
8		3	DEPARTMENT	N	3.0				AG	53	35
9		3	SEQ-NR	N	2.0				AH N	56	38
10		3	CHECKDIGIT	N	2.0				AI N	58	3A
11		2	SPECIALIST	A	30.0			A	AJ N	60	3C
12		2	REC-TYPE	N	2.0				MI N	90	5A
13	SP	1	ART-KEY	A	16.0			P	SK	44	2C
** *** VARIABLE PART ***											
14		1	COMPONENTS-NR	N	2.0				AS N	92	5C
15	PE	1	COMPONENTS					99	AT	94	5E
16		2	COMPONENT-ART-NR	A	12.0				AU N	94	5E
17		2	COMPONENT-QUANTITY	P	5.0				AV	106	6A
18		1	STOCKS-CNT	N	2.0				B2 N	1579	62B
19	PE	1	STOCKS					30	AZ	1581	62D
20		2	STOCK-NR	A	12.0				A0 N	1581	62D
21		2	STOCK-QUANTITY	I	4.0				A1 F	1593	639
*** TOTAL LENGTH ***										2061	80D
*** End of report ***											

```

13:45:17          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - List File with Children -

File ID ..... AZ-QA-AVB-L-ART-SALE
Type ..... Logical VSAM
-----

```

Cnt	Ty	L	Field ID	F	Length	Occ	D	DB	S	Off Dec	Off Hex
** *** FIXED PART ***											
1	GR	1	ART-FIX					AA		0	0
2		2	ART-DESC	A	32.0			A	AB N	0	0
3		2	DATE-ADD	N	6.0			AC	N	32	20
4		2	DATE-MODIFIED	N	6.0			AD	N	38	26
5	GR	2	ART-NR					AE		44	2C
6		3	ART-REC-GROUP	A	5.0			II	N	44	2C
7		3	ART-GROUP	A	4.0			AF	N	49	31
8		3	DEPARTMENT	N	3.0			AG		53	35
9		3	SEQ-NR	N	2.0			AH	N	56	38
10		3	CHECKDIGIT	N	2.0			AI	N	58	3A
11		2	SPECIALIST	A	30.0			A	AJ N	60	3C
12		2	REC-TYPE	N	2.0			MI	N	90	5A
13	RE	1	ART-FIX							0	0
14		2	DEMO1	A	22.0					0	0
15	PE	2	DEMO-PE					1		22	16
16		3	DEMO2	A	22.0					22	16
17	RE	1	ART-FIX							0	0
18	PE	2	DEMO-PE2					1		0	0
19		3	DEMO3	A	44.0					0	0
20	SP	1	ART-KEY	A	16.0			P	SK	44	2C
>>> WARNING: VSAM SP-FIELD DEFINED BEFORE SOURCE-FIELD. <<<											
21		1	PRICE	N	5.2			AM	N	92	5C
*/ *** VARIABLE PART ***											
22	MU	1	DISCOUNT	N	5.2			6	AN N	99	63
23	PE	1	STATISTICS-MONTH					12	AP	141	8D
24		2	TURNOVER	N	5.0				AQ N	141	8D
25		2	QUANTITY	N	5.0				AR N	146	92
26		1	STOCKS-NR-DELIVERY-CNT	N	2.0				B1 N	261	105
27	MU	1	STOCKS-NR-DELIVERY	A	12.0			20	AO N	263	107
*** TOTAL LENGTH ***										503	1F7
*** End of report ***											

Sample Output

```

13:40:20          ***** P R E D I C T 4.3.1 *****                2003-05-31
                    - Generate Transparency Table -                    Page:    1
File: AZ-QA-AVB-V-ART                                         Fnr:
126

                    *** Table of Keys ***

                    Type  Offset  Length  DD name  Unique
                    P      44     16     AVBVART   Y
                    A       0     32     ARTDESC
                    A      60     30     ARTSPEC

                    *** Sorted table of logical VSAM files ***

Indicator - Offset:   44  Length:    5  Number of logical VSAM files:   2

156 AZ-QA-AVB-L-ART-COMP                                IND-V: 5x3b1
155 AZ-QA-AVB-L-ART-SALE                                IND-V: 12345
    
```

```

13:40:44          ***** P R E D I C T 4.3.1 *****                2003-05-31
                    - Generate Transparency Table -                    Page:    2
File: AZ-QA-AVB-A-ART                                         Fnr:    76
T  L              Adabas Field          F  Length  D DB Occ      File name
                    Corresponding VSAM Field(s)
-----
GR 1 ART-FIX                                     AA
GR 1 ART-FIX                                     AZ-QA-AVB-L-ART-COMP
GR 1 ART-FIX                                     AZ-QA-AVB-L-ART-SALE

2 ART-DESC                                     A   32.0  D AB
2 ART-DESC                                     A   32.0  A   AZ-QA-AVB-L-ART-COMP
2 ART-DESC                                     A   32.0  A   AZ-QA-AVB-L-ART-SALE

2 DATE-ADD                                     N    6.0   AC
2 DATE-ADD                                     N    6.0
2 DATE-ADD                                     N    6.0   AZ-QA-AVB-L-ART-COMP
2 DATE-ADD                                     N    6.0   AZ-QA-AVB-L-ART-SALE

2 DATE-MODIFIED                               N    6.0   AD
2 DATE-MODIFIED                               N    6.0   AZ-QA-AVB-L-ART-COMP
2 DATE-MODIFIED                               N    6.0   AZ-QA-AVB-L-ART-SALE
GR 2 ART-NR                                     AE
GR 2 ART-NR                                     AZ-QA-AVB-L-ART-COMP
GR 2 ART-NR                                     AZ-QA-AVB-L-ART-SALE

3 ART-REC-KEY                                  A    5.0   II
3 ART-REC-KEY                                  A    5.0   AZ-QA-AVB-L-ART-COMP
3 ART-REC-GROUP                                A    5.0   AZ-QA-AVB-L-ART-SALE
    
```

```

13:40:44          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate Transparency Table -          Page: 3
File: AZ-QA-AVB-A-ART          Fnr: 76
T L          Adabas Field          F Length D DB Occ          File name
      Corresponding VSAM Field(s)
-----
3 ART-GROUP          A      4.0 D AF          AZ-QA-AVB-L-ART-COMP
3 ART-GROUP          A      4.0          AZ-QA-AVB-L-ART-SALE
3 ART-GROUP          A      4.0          AZ-QA-AVB-L-ART-SALE

3 DEPARTMENT        N      3.0 AG          AZ-QA-AVB-L-ART-COMP
3 DEPARTMENT        N      3.0          AZ-QA-AVB-L-ART-SALE
3 DEPARTMENT        N      3.0          AZ-QA-AVB-L-ART-SALE

3 SEQ-NR             N      2.0 AH          AZ-QA-AVB-L-ART-COMP
3 SEQ-NR             N      2.0          AZ-QA-AVB-L-ART-SALE
3 SEQ-NR             N      2.0          AZ-QA-AVB-L-ART-SALE

3 CHECKDIGIT        N      2.0 AI          AZ-QA-AVB-L-ART-COMP
3 CHECKDIGIT        N      2.0          AZ-QA-AVB-L-ART-SALE
3 CHECKDIGIT        N      2.0          AZ-QA-AVB-L-ART-SALE
3 DEPARTMENT        N      3.0 AG          AZ-QA-AVB-L-ART-COMP
3 DEPARTMENT        N      3.0          AZ-QA-AVB-L-ART-SALE
3 DEPARTMENT        N      3.0          AZ-QA-AVB-L-ART-SALE

3 SEQ-NR             N      2.0 AH          AZ-QA-AVB-L-ART-COMP
3 SEQ-NR             N      2.0          AZ-QA-AVB-L-ART-SALE
3 SEQ-NR             N      2.0          AZ-QA-AVB-L-ART-SALE

3 CHECKDIGIT        N      2.0 AI          AZ-QA-AVB-L-ART-COMP
3 CHECKDIGIT        N      2.0          AZ-QA-AVB-L-ART-SALE
3 CHECKDIGIT        N      2.0          AZ-QA-AVB-L-ART-SALE

2 SPECIALIST        A     30.0 D AJ          AZ-QA-AVB-L-ART-COMP
2 SPECIALIST        A     30.0 A          AZ-QA-AVB-L-ART-SALE
2 SPECIALIST        A     30.0 A          AZ-QA-AVB-L-ART-SALE

2 REC-TYPE          N      2.0 D MI          AZ-QA-AVB-L-ART-COMP
2 REC-TYPE          N      2.0          AZ-QA-AVB-L-ART-SALE
2 REC-TYPE          N      2.0          AZ-QA-AVB-L-ART-SALE

SP 1 ART-KEY        A     16.0 D SK          AZ-QA-AVB-L-ART-COMP
SP 1 ART-KEY        A     16.0 P          AZ-QA-AVB-L-ART-SALE
SP 1 ART-KEY        A     16.0 P          AZ-QA-AVB-L-ART-SALE
    
```

```

13:40:44          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate Transparency Table -          Page: 4
File: AZ-QA-AVB-A-ART          Fnr: 76
T L          Adabas Field          F Length D DB Occ          File name
      Corresponding VSAM Field(s)
-----
      1 PRICE          N      5.2  AM
      1 PRICE          N      5.2          AZ-QA-AVB-L-ART-SALE
MU 1 DISCOUNT          N      5.2  AN 6
MU 1 DISCOUNT          N      5.2  6 AZ-QA-AVB-L-ART-SALE
      1 STOCKS-NR-DELIVERY-CNT          N      2.0  B1
      1 STOCKS-NR-DELIVERY-CNT          N      2.0          AZ-QA-AVB-L-ART-SALE
MU 1 STOCKS-NR-DELIVERY          A     12.0  D AO 20
MU 1 STOCKS-NR-DELIVERY          A     12.0          20 AZ-QA-AVB-L-ART-SALE
PE 1 STATISTICS-MONTH          AP 12
PE 1 STATISTICS-MONTH          12 AZ-QA-AVB-L-ART-SALE
      2 TURNOVER          N      5.0  AQ
      2 TURNOVER          N      5.0          AZ-QA-AVB-L-ART-SALE
      2 QUANTITY          N      5.0  AR
      2 QUANTITY          N      5.0          AZ-QA-AVB-L-ART-SALE
      1 COMPONENTS-NR          N      2.0  AS
      1 COMPONENTS-NR          N      2.0          AZ-QA-AVB-L-ART-COMP
    
```

```

13:40:44          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate Transparency Table -          Page: 5
File: AZ-QA-AVB-A-ART          Fnr: 76
T L          Adabas Field          F Length D DB Occ          File name
    Corresponding VSAM Field(s)
-----
PE 1 COMPONENTS          AT 64
PE 1 COMPONENTS          99 AZ-QA-AVB-L-ART-COMP
    >>> Check different field occurrences <<<

    2 COMPONENT-ART-NR          A 12.0 AU
    2 COMPONENT-ART-NR          A 12.0 AZ-QA-AVB-L-ART-COMP

    2 COMPONENT-QUANTITY          P 5.0 AV
    2 COMPONENT-QUANTITY          P 5.0 AZ-QA-AVB-L-ART-COMP

PE 1 COMPONENTS-C          AW 35

    2 COMPONENT-ART-NR-C          A 12.0 AX
    2 COMPONENT-QUANTITY-C          P 5.0 AY

    >>> Splitting of VSAM tables supposed. <<<

    1 STOCKS-CNT          N 2.0 B2
    1 STOCKS-CNT          N 2.0 AZ-QA-AVB-L-ART-COMP

PE 1 STOCKS          AZ 30
PE 1 STOCKS          30 AZ-QA-AVB-L-ART-COMP

    2 STOCK-NR          A 12.0 A0
    2 STOCK-NR          A 12.0 AZ-QA-AVB-L-ART-COMP

    2 STOCK-QUANTITY          I 4.0 A1
    2 STOCK-QUANTITY          I 4.0 AZ-QA-AVB-L-ART-COMP
    
```

COMPONENTS												
13:48:07		***** P R E D I C T 4.3.1 *****					2003-05-31					
		- Generate Transparency Table -					Page: 6					
File: AZ-QA-AVB-L-ART-COMP							Fnr: 156					
CNT	T	L	VSAM field	F	Length	D	DB	Occ	OFF	FB	OFF	
		Corresponding Adabas field										

1	GR	1	ART-FIX ART-FIX				AA		0	AA	0	
2		2	ART-DESC ART-DESC	A	32.0		A AB D		0	AA	0	
3		2	DATE-ADD DATE-ADD	N	6.0		AC		32	AA	32	
4		2	DATE-MODIFIED DATE-MODIFIED	N	6.0		AD		38	AA	38	
5	GR	2	ART-NR ART-NR				AE		44	AA	44	
6		3	ART-REC-KEY ART-REC-KEY	A	5.0		II		44	AA	44	
7		3	ART-GROUP ART-GROUP	A	4.0		AF D		49	AA	49	
8		3	DEPARTMENT DEPARTMENT	N	3.0		AG		53	AA	53	
9		3	SEQ-NR SEQ-NR	N	2.0		AH		56	AA	56	
10		3	CHECKDIGIT CHECKDIGIT	N	2.0		AI		58	AA	58	
11		2	SPECIALIST SPECIALIST	A	30.0		A AJ D		60	AA	60	
12		2	REC-TYPE REC-TYPE	N	2.0		MI D		90	AA	90	
13	SP	1	ART-KEY ART-KEY CHECKDIGIT	A	16.0		P SK D		44			
11		2	SPECIALIST SPECIALIST	A	30.0		A AJ D		60	AA	60	
12		2	REC-TYPE REC-TYPE	N	2.0		MI D		90	AA	90	
13	SP	1	ART-KEY ART-KEY	A	16.0		P SK D		44			
14		1	COMPONENTS-NR COMPONENTS-NR	N	2.0		AS		92	AS	0	
15	PE	1	COMPONENTS				AT	99	94	AT	0	

```

13:48:07          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate Transparency Table -          Page: 7
File: AZ-QA-AVB-L-ART-COMP          Fnr: 156
CNT T L          VSAM field          F Length D DB Occ OFF FB OFF
                    Corresponding Adabas field          DB FB
-----
                    >>>> Table is splitted in the Adabas file <<<<

16      2 COMPONENT-ART-NR          A    12.0   AU    94    AT 0
          COMPONENT-ART-NR

17      2 COMPONENT-QUANTITY        P     5.0   AV   106    AT 12
          COMPONENT-QUANTITY

18      1 STOCKS-CNT                N     2.0   B2   1579   B2 0
          STOCKS-CNT

19      PE 1 STOCKS                  AZ   30 1581   AZ 0
          STOCKS

20      2 STOCK-NR                  A    12.0   A0   1581   AZ 0
          STOCK-NR

21      2 STOCK-QUANTITY            I     4.0   A1   1593   AZ 12
          STOCK-QUANTITY
    
```

```

13:48:07          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate Transparency Table -          Page: 8
File: AZ-QA-AVB-L-ART-SALE          Fnr: 155
CNT T L          VSAM field          F Length D DB Occ OFF FB OFF
                    Corresponding ADABAS field          DB FB
-----
1  GR 1 ART-FIX          AA 0 AA 0
    ART-FIX
2  2 ART-DESC          A 32.0 A AB 0 AA 0
    ART-DESC          D
3  2 DATE-ADD          N 6.0 AC 32 AA 32
    DATE-ADD
4  2 DATE-MODIFIED          N 6.0 AD 38 AA 38
    DATE-MODIFIED
5  GR 2 ART-NR          AE 44 AA 44
    ART-NR
6  3 ART-REC-GROUP          A 5.0 II 44 AA 44
    ART-REC-KEY
7  3 ART-GROUP          A 4.0 AF 49 AA 49
    ART-GROUP          D
8  3 DEPARTMENT          N 3.0 AG 53 AA 53
    DEPARTMENT
9  3 SEQ-NR          N 2.0 AH 56 AA 56
    SEQ-NR
10 3 CHECKDIGIT          N 2.0 AI 58 AA 58
    CHECKDIGIT
11 2 SPECIALIST          A 30.0 A AJ 60 AA 60
    SPECIALIST          D
12 2 REC-TYPE          N 2.0 MI 90 AA 90
    REC-TYPE          D
    
```

```

13:48:07          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate Transparency Table -          Page: 9
File: AZ-QA-AVB-L-ART-SALE          Fnr: 155
CNT T L          VSAM field          F Length D DB Occ OFF FB OFF
                    Corresponding ADABAS field          DB FB
-----
13 SP 1 ART-KEY          A 16.0 P SK 44
                    ART-KEY          D
10 3 CHECKDIGIT          N 2.0 AI 58 AA 58
                    CHECKDIGIT
11 2 SPECIALIST          A 30.0 A AJ 60 AA 60
                    SPECIALIST          D
12 2 REC-TYPE          N 2.0 MI 90 AA 90
                    REC-TYPE          D
13 SP 1 ART-KEY          A 16.0 P SK 44
                    ART-KEY          D
14 1 PRICE          N 5.2 AM 92 AM 0
                    PRICE
15 MU 1 DISCOUNT          N 5.2 AN 6 99 AN 0
                    DISCOUNT
16 PE 1 STATISTICS-MONTH          AP 12 141 AP 0
                    STATISTICS-MONTH
17 2 TURNOVER          N 5.0 AQ 141 AP 0
                    TURNOVER
    
```

```

13:48:07          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate Transparency Table -          Page: 10
File: AZ-QA-AVB-L-ART-SALE          Fnr: 155
CNT T L          VSAM field          F Length D DB Occ OFF FB OFF
                    Corresponding ADABAS field          DB FB
-----
18 2 QUANTITY          N 5.0 AR 146 AP 5
                    QUANTITY
19 1 STOCKS-NR-DELIVERY-CNT          N 2.0 B1 261 B1 0
                    STOCKS-NR-DELIVERY-CNT
20 MU 1 STOCKS-NR-DELIVERY          A 12.0 AO 20 263 AO 0
                    STOCKS-NR-DELIVERY          D
    
```

```

13:48:07          ***** P R E D I C T  4.3.1  *****          2003-05-31
                    - Generate Transparency Table -                      Page:   11
File: AZ-QA-AVB-V-ART                                     Fnr:   126
*****
*   THIS TRANSPARENCY TABLE WAS GENERATED BY PREDICT
*   FOR VSAM   FILE: AZ-QA-AVB-V-ART
*   ADABAS FILE: AZ-QA-AVB-A-ART
*   ON: 2003-05-31 AT 13:48:07
*****
MCTAB TYPE=GEN, FN=AVBVART, FNR=76, RECSIZ=2061, -
      KEY1=SK, KEYLEN=16, KEYOFF=44, INDXTYP=P, -
      ODNAME=(AZ, AO), -
      ODLEN=(2, 2), -
      ODFMT=(U, U), OSOFF=(1581, 263), -
      ODOFF=(1579, 261), ODPOS=(0, 0), OSSIZ=(16, 12), -
      RECTYPE=Y, RECIDFB=II, RECTBYT=5, RECTOFF=44, RECTCNT=2, -
      RECTFMT=A, -
      RECFMTS=( '5x3b1AA, AS, AT1-64, AW1-35, B2, AZ001-030', '12345A-
A, AM, AN1-6, AP1-12, B1, AO001-020' )
MCTAB TYPE=GEN, FN=ARTDESC, FNR=76, RECSIZ=2061, -
      KEY1=AB, KEYLEN=32, KEYOFF=0, INDXTYP=S, UNIQUE=N, -
      ODNAME=(AZ, AO), -
      ODLEN=(2, 2), -
      A, AM, AN1-6, AP1-12, B1, AO001-020' )
MCTAB TYPE=GEN, FN=ARTDESC, FNR=76, RECSIZ=2061, -
      KEY1=AB, KEYLEN=32, KEYOFF=0, INDXTYP=S, UNIQUE=N, -
      ODNAME=(AZ, AO), -
      ODLEN=(2, 2), -
      ODFMT=(U, U), OSOFF=(1581, 263), -
      ODOFF=(1579, 261), ODPOS=(0, 0), OSSIZ=(16, 12), -
      RECTYPE=Y, RECIDFB=II, RECTBYT=5, RECTOFF=44, RECTCNT=2, -
      RECTFMT=A, -
      RECFMTS=( '5x3b1AA, AS, AT1-64, AW1-35, B2, AZ001-030', '12345A-
A, AM, AN1-6, AP1-12, B1, AO001-020' )
MCTAB TYPE=GEN, FN=ARTSPEC, FNR=76, RECSIZ=2061, -
      KEY1=AJ, KEYLEN=30, KEYOFF=60, INDXTYP=S, UNIQUE=N, -
      ODNAME=(AZ, AO), -
      ODLEN=(2, 2), -
      ODFMT=(U, U), OSOFF=(1581, 263), -
      ODOFF=(1579, 261), ODPOS=(0, 0), OSSIZ=(16, 12), -
      RECTYPE=Y, RECIDFB=II, RECTBYT=5, RECTOFF=44, RECTCNT=2, -
      RECTFMT=A, -
      RECFMTS=( '5x3b1AA, AS, AT1-64, AW1-35, B2, AZ001-030', '12345A-
A, AM, AN1-6, AP1-12, B1, AO001-020' )

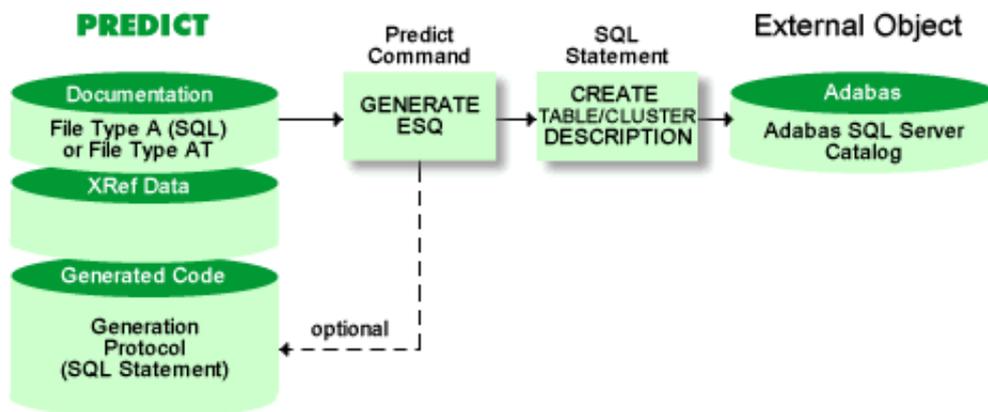
*** End of transparency table ***

```

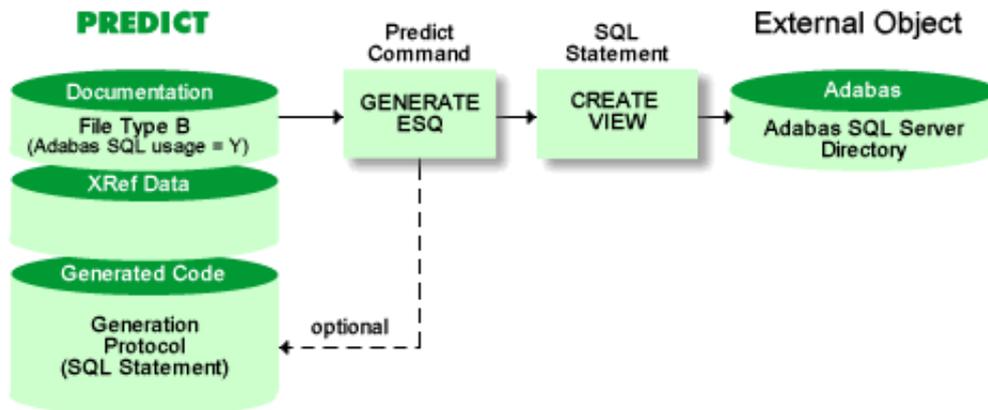
Adabas Table/Cluster/View

This function is used to create table descriptions, cluster descriptions or views in an Adabas SQL Server catalog.

Predict Object(s)	Generated Command	Result
One Adabas file corresponding to one SQL table	CREATE TABLE DESCRIPTION	Adds a description of a table in an Adabas SQL Server catalog without creating an Adabas file.
One AT file corresponding to one SQL table		
Multiple AT files corresponding to multiple SQL tables	CREATE CLUSTER DESCRIPTION	Adds description of several base tables containing multiple value fields or periodic groups with variable occurrences in Adabas SQL Server.



From files of type B, a CREATE VIEW statement is generated.



See also section Adabas SQL Server in the **Predict and Other Systems** documentation.

This section contains:

- Calling the Function
- Recommendations when Using Adabas Vista
- Generate Adabas Table/View in Batch Mode

Calling the Function

Display the Generate Adabas table/view screen with function code G and object code EQ in a Predict main menu, or with the command GENERATE ESQ.

```

09:33:43          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan   0          - Generate Adabas table/view -

Current VM ..... HOME
File ID ..... *
Contained in DA ..... *

Add table / view ..... Y (Y,N)          Use Vista access-nr ..* Y
Replace table / view ..... N (Y,N)
Drop table / view cascade .. N (Y,N)

List SQL statements ..... Y (Y,N)

Protocol saved in member ... *
                in library .. ASQLIB

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last Lnke1 Flip Print Impl AdmFi Selfi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presettings	
The parameters below can be changed in the Modify Adabas Table/View Defaults screen. See Generation Defaults.	
Specification DB ID	<p>Y Database ID must be specified.</p> <p>N Database ID may be specified.</p> <p>This parameter is set to N at installation.</p>
Count of saved actions	Number (0-99) of generation actions which are stored in the protocol specified under Protocol saved in member/library. If 99 is entered, all generation actions are stored. This is the default setting at installation.
Parameters	

Current VM	Adabas tables/views can only be generated from Predict file objects contained in databases of type local linked to the Current VM or contained in databases accessible via the network containing the Current VM. This is a read-only field.
File ID	<p>ID of the Predict file object from which the definitions are to be generated. To generate an Adabas table, the file must be of</p> <ul style="list-style-type: none"> ● type A (with parameter SQL usage set to Y) or ● type AT. <p>To generate an Adabas view, the file must be of type B. To generate an Adabas cluster, the file must be of type AT. Enter only one file from the cluster; all files in the cluster will be used for generation.</p>
Contained in DA	<p>Only applicable to tables. Depending on the setting of Specification DB ID in the Modify Adabas table/view defaults screen, entry of a Database ID is mandatory or optional for Adabas files and userviews. If a database is specified, its file list must contain the specified file and the Database type must be compatible with the database type specified for this database number within the Natural Parameter Module (via NTDB macro). Enter an asterisk to display a list of databases for selection.</p>
Use Vista access-nr	<p>N Vista access number is not used.</p> <p>T Vista access number is used and the system tests whether a Vista element for this file with the same number exists. If so, and if no database ID is specified and the file contains a Vista access number, an Adabas table/view is generated with the access number taken from the file. If the file has no Vista access number, an error message is given.</p> <p>Y Vista access number is used, but no testing is performed. If no database ID is specified and the file contains a Vista access number, an Adabas table/view is generated with the access number taken from the file. If the file has no Vista access number, an error message is given.</p>
Add table/view	<p>Y A new Adabas table, view or cluster is created.</p> <p>If the external object does not exist, this parameter must be set to Y.</p>
Replace table/view	<p>Y If the Adabas table or view already exists, an Adabas statement DROP TABLE DESCRIPTION or DROP VIEW is generated first.</p> <p>If a cluster already exists for one of the files in the cluster structure, a DROP CLUSTER DESCRIPTION statement is generated.</p>

Drop table/view cascade	<p>Y</p> <p>For tables: Tables and all dependent views will be dropped. The statement DROP TABLE DESCRIPTION <file-ID> CASCADE is generated first.</p> <p>For clusters: All tables in the cluster will be dropped. The statement DROP CLUSTER DESCRIPTION <cluster-name> CASCADE is generated first. The cluster-name is taken from the catalog.</p> <p>Note: If the table already exists and Replace table/view and Drop table/view cascade are set to Y, the dependent views are listed. If the purge is confirmed, the table and all dependent views will be deleted and only the table will be generated. Dependent views must then be generated separately. For clusters, all base tables which belong to the cluster and all dependent views are listed. If the purge is confirmed, all listed objects will be deleted.</p>
List SQL statements	<p>Y</p> <p>Generated SQL statements are listed.</p>
Protocol saved in member	<p>SQL statements generated for the implementation of the database can be saved in a Natural source member (protocol) on the FDIC file.</p> <p>A member name for the protocol is only required when the Adabas table/view is generated for the first time. SQL statements generated in subsequent generation runs are concatenated to the protocol.</p> <p>The protocol can be displayed using the Display function of the Administration Implemented File menu.</p>
Protocol saved in library	<p>Library where the protocol is to be saved.</p> <p>Note: All tables in a cluster are given the same member/library name.</p>

Recommendations when Using Adabas Vista

If using Predict with Vista distributed data processing, we recommend the following:

- Always enter a Vista access number for Adabas files (Add, Modify file).
- Set the parameter Specification DB ID in the Modify Adabas table/view defaults to N (database ID may be specified but is not mandatory).
If this parameter is set to Y, the user is forced to enter a database ID when generating an Adabas table/view from an Adabas file.
- Set the parameter Use Vista access-nr in the Modify Adabas table/view defaults screen to T, but do not protect the field.

If you do not want to use Vista:

- Set the parameter Use Vista access-nr in the Modify Adabas table/view defaults screen to N and deactivate the option by blanking out the preceding X.

The table below illustrates the use of the parameter Use Vista access-nr.

Specification DB ID	Use Vista Access Number	Result if Database ID is specified
---------------------	-------------------------	------------------------------------

Y	N, Y, T	<p>[1] If the file is Simple in this database and the Adabas attribute Vista access only is not set to Y: the Adabas table/view is generated with the physical database number and physical file number.</p>
Y	N	<p>[2] If the file is defined as partitioned in this database or the Adabas attribute Vista access only=Y: the system checks whether a Vista element for this file and database exists.</p> <p>If exactly one element exists, the Adabas table is generated with the logical Vista number defined in this Vista element.</p> <p>If the Vista numbers are used, the CREATE TABLE DESCRIPTION statement is appended by 'MODIFICATION NOT ALLOWED' and has the result that an ALTER statement for this table is rejected in Adabas.</p> <p>If more than one Vista element is found, no Adabas table is generated and an error message is given.</p>
Y	T,Y	<p>Same as above, except that if several Vista elements exist for the file and database, the system checks whether the Vista access number of the file is identical to the Vista file number of one of these Vista elements.</p> <p>If identical, the Adabas table is generated.</p> <p>If not, no Adabas table is generated and an error message is given.</p> <p>If the Vista numbers are used, the CREATE TABLE DESCRIPTION statement is appended by 'MODIFICATION NOT ALLOWED' and has the result that an ALTER statement for this table is rejected in Adabas.</p>
N	Y	<p>If the file does not contain Vista access numbers, an error message is given. If the file does contain Vista access numbers, an Adabas table is generated with the access number taken from the file.</p> <p>If the Vista numbers are used, the CREATE TABLE DESCRIPTION statement is appended by 'MODIFICATION NOT ALLOWED' and has the result that an ALTER statement for this table is rejected in Adabas.</p>
N	Y	<p>If the file does not contain Vista access numbers, an error message is given. If the file does contain Vista access numbers, the system tests whether Vista elements with the same numbers exist for this file. If so, an Adabas table is generated with this access number.</p> <p>If the Vista numbers are used, the CREATE TABLE DESCRIPTION statement is appended by 'MODIFICATION NOT ALLOWED' and has the result that an ALTER statement for this table is rejected in Adabas.</p>
N	Y	<p>If the file does not contain Vista access numbers, an error message is given. If the file does contain Vista access numbers, the system tests whether a Vista element with the same number exists for this file. If so, an Adabas table is generated with this access number.</p> <p>If the Vista numbers are used, the CREATE TABLE DESCRIPTION statement is appended by 'MODIFICATION NOT ALLOWED' and has the result that an ALTER statement for this table is rejected in Adabas.</p>

Generate Adabas Table/View in Batch Mode

Command: GENERATE ESQ

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

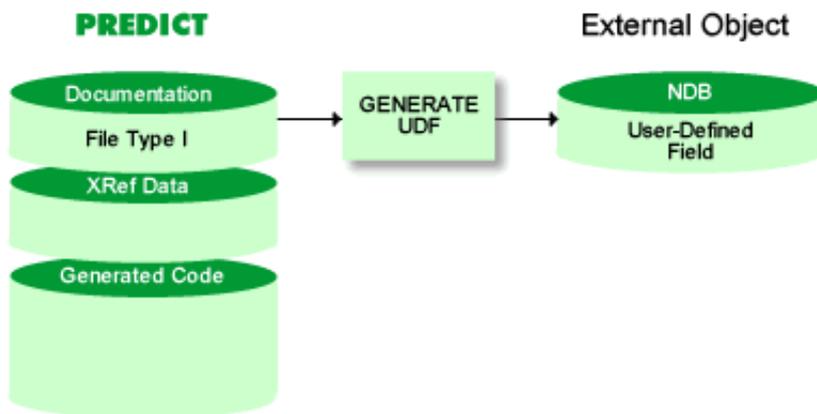
Field	Keyword	Position
File ID	FILE-ID	1
Database ID	DATABASE-ID	2
Add table/view	ADD-TABLE	3
Replace table/view	REPLACE	4
Drop table/view cascade	CASCADE	5
List Adabas statement	LIST	6
Protocol saved in member	MEM	7
Protocol saved in library	LIB	8
Use Vista access-nr	USE-ACCESS-NR	9

Generate for Natural

The following types of external objects can be generated for Natural:

- IMS User-Defined Fields
- Replace Verification Rule
- Data Definition Module

IMS User-Defined Fields



IMS UDFs are required to access data in IMS segments from Natural for DL/I programs. UDFs are used for mapping IMS segments to fields in DDMs.

IMS UDFs for physical or logical IMS segments can be generated from Predict file objects of type I. IMS UDFs contain the segment name and - in the case of physical IMS segments - the user-defined field definitions for one IMS segment.

When generating IMS UDFs for logical IMS segments, the IMS UDFs for physical IMS segments that the logical IMS segments are derived from are also generated if necessary (if they do not already exist or if the Predict file objects for the physical IMS segments have been modified since generation).

Rules Applying to the Generation of IMS UDFs

- In a first step the Generate IMS User-Defined Fields function collects file objects of the types I, K or J that are affected by the operation and displays them in a table. See Displaying all Predict file objects affected by UDF Generation.
- The offsets of the IMS descriptor fields in the IMS segment layout (file type J) are checked for consistency with the respective descriptor fields in IMS segments (file type I) before an IMS UDF is actually generated.
- A database and file number is assigned to the newly generated IMS UDF. DDMs use this database number and file number.
- If differences between the format of IMS descriptor fields in the NDB and the respective format definitions of Predict are detected, the IMS descriptor field formats in the NDB are changed. Differences can occur if the descriptor fields have the format packed or packed-signed.

Calling the Function

The Generate IMS UDF screen is displayed with function code G and object code UD in a Predict main menu, or with the command GENERATE UDF

```

13:45:33          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Generate IMS UDF -

File ID .....*
Contained in DA ...

Overwrite option .. Y  (Y,N)
List IMS segment .. Y  (Y,N)
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Parameters	
File ID	ID of the Predict file object from which UDF is to be generated. The file must be of type I. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.
Contained in DA	ID of the Predict database containing the file. Must be of type I and the file list of the database must contain the file.
Overwrite option	Y Existing UDFs are overwritten by the newly generated UDFs.
List IMS segment	Y The generated fields are to be listed.

Displaying all Predict File Objects Affected by UDF Generation

Before actually generating IMS UDFs, Predict checks which file objects of the types I, K or J are affected and displays the results of this check in a table.

```

13:00:32          ***** P R E D I C T  4.3.1  *****          2003-05-31
                                - Checked IMS files -
                                File Segm.
File name                    Type  Type  Relation      Udf      DBnr/Fnr
PARTUSED-LCONTD              I    L    entered file  nonexist
ARTICLE-CONTAIND            I    V    virt.child
ARTICLE-PART                 I    P    log.parent    nonexist
ARTICLE-CONTAINS            I    C    real.child    nonexist
    
```

Columns in the Table		
File name	ID of a Predict file object.	
File Type	Predict file type.	
Segment Type	Type of IMS segment.	
Relation	How the Predict file object is related to the file to which the generation function is applied	
	entered file	File to which the generation function is applied.
	phys. file	For a logical segment, the physical segment from which it is derived.
	log child	For a concatenated segment derived from a logical child and its logical parent, the logical child from which it is derived.
	log. parent	For a concatenated segment, the parent segment from which it is derived.
	virtual child	For a concatenated segment derived from a virtual logical child and the physical parent of its paired real logical child, the virtual logical child from which it is derived.
	real child	For a virtual logical child segment the real logical child segment from which it is derived.
UDF	Status of the UDF	
	generated	UDF exists for the file
	modified	The Predict file/field objects have been modified after generation of the UDF.
	nonexist	UDF does not exist
	invalid	The generated UDF points to a wrong DBNR, i.e. the database number is not defined as IMS or DL/I database number in the Natural parameter module via the NTDB macro
	not found	The generated UDF is deleted
	UDF diff.	The segment in the UDF differs from the segment name in the file, i.e. the generated segment was modified with an external utility.
DBnr/Fnr	Database and file number assigned to the UDF. Note: If a DDM for Natural IMS or DL/I is to be generated, all related IMS UDFs must be of status generated.	

Generate IMS User-Defined Fields in Batch Mode

Command: GENERATE UDF

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Contained in DA	DATABASE-ID	2
Overwrite option	REPLACE	3
List IMS segment	LIST	4

Sample Output

```

13:29:33          ***** P R E D I C T 4.3.1 *****          2003-05-31
                        - List IMS UDFs -                          Page: 1
Database: PARTUSED          DBD:
File   : PARTUSED-LCONTD    IMS Segment: LCONTD
                               Seg. length: 600-700

  T  L DB Field name          F   Length  D  Occ  Offset Var
  ---
  1  NG PARTNO                N     3.0  SRC      201
  1  NH PARTNAME              A    20.0  SRC      291
  1  NI XPART                 A    20.0  AIX      200
**** IMS udf generated ( Dbnr: 153 Fnr: 84 ) ****
    
```

Replace Verification Rule

The Replace Verification Rule function can be applied only to rules of status automatic. A rule is assigned status automatic by applying the function Generate DDM to the file containing the field that the corresponding Verification object is linked to via *Is verified by VE*.

The function Replace Verification Rule replaces the code of the rule; all links to fields remain unchanged.

Calling the Function

The Replace Verification Rule screen is displayed with function code G and object code RU in a Predict main menu, or with the command GENERATE RULE.

```

09:40:26          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan   0          - Replace Verification Rule -

Rule ID .....

List related fields ... Y (Y,N)
List related maps .... Y (Y,N)
List generated rule ... Y (Y,N)
    
```

Values for input fields which have been locked by the data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Parameters	
Rule ID	ID of the Predict verification object from which the verification rule is to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.
List related fields	Y The IDs of all fields connected to this verification rule via <i>Is verified by VE</i> are to be listed.
List related maps	Natural maps using replaced verifications are to be recataloged so that they use the new verification rules. List related maps helps to find these maps. Y The IDs of all maps connected to this verification rule are to be listed.
List generated code	Y The generated code is to be listed.

Replace Verification Rule in Batch Mode

Command: GENERATE RULE

Enter parameters on next line in positional or keyword form. Rule ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
Rule ID	RULE-ID	1
List related fields	RELATE	2
List generated code	LIST	3
List related maps	RELATED-MAPS	4

Sample Output

```

13:04:07          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Replace Verification Rule -                   Page: 1

      Link of Verification GEN-CHECK-ZERO to Elementary Fields
      File Name                               Field Name                               Remark
-----
GENERATION-EXAMPLE                           ELE-N-9V5                               activ
                                                ELE-PS-5V2                               activ
                                                PC-MC-PS-6V1                             activ
Gen-examples                                 Ele-N-9V5                               not activ
                                                Ele-PS-5V2                               not activ
                                                PC-MU-PS-6V1                             not activ
    
```

```

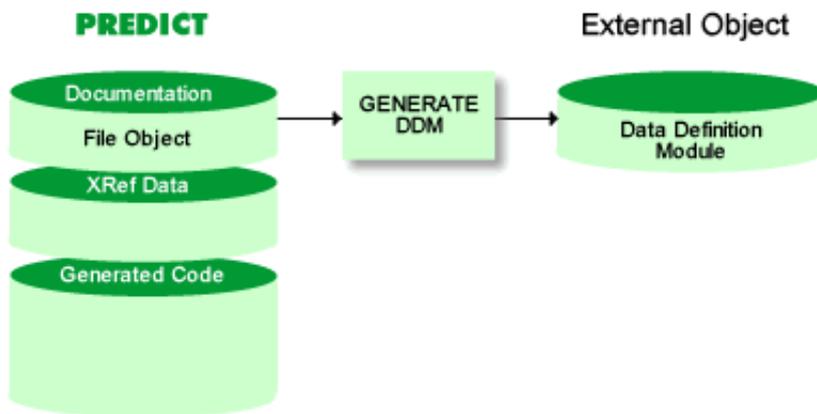
13:04:07          ***** P R E D I C T  4.3.1  *****          2003-05-31
                    - Replace Verification Rule -                    Page:    2

Rule: GEN-CHECK-ZERO

0010 * *****
0020 * Verification: GEN-CHECK-ZERO generated by Predict          *
0030 * with format: Numeric; Type: Not Equal;                    *
0040 *           on: 2003-05-31; at: 13:03:46; from user: SMR;    *
0050 * *****
0060 IF & = 0
0070 REINPUT 'Invalid value(s): ''0'' .'
0080           MARK *&

```

Data Definition Module



Rules Applying to the Generation of DDMs

- If Natural Security is installed and the specified file is defined to Natural Security, the same checks are performed as are performed before a DDM can be regenerated or a file can be added. See description of the parameter DDM Modifier in the section **File Maintenance** of the **Natural Security documentation**.
- A DDM generated for a physically coupled file contains the ID of the file to which it is coupled and the short names of the fields via which they are coupled as a comment. For example: "Coupled via AA to GA of FINANCE".
- The names of the Predict file and its userviews begin with SYSDIC. When a DDM is generated for a file whose name begins with SYSDIC, the logical database number (255) and logical file number (253) that point to the FDIC are taken.
- If the file (which has a DDM) was renamed, the old DDM is purged in the Generate DDM function and the new DDM is added. If Natural Security objects or Super Natural objects exist for the old DDM, these objects are renamed too.

Note:

Rules applying to the generation of DDMs for use with IMS or Adabas Vista are described in the sections **Generating DDMs for Use with IMS** and **Generating a DDM for Use with Adabas Vista**.

When generating DDMs on Windows or UNIX platforms and the FDIC file is located on the mainframe, the DDM will not be available on the mainframe. It will be stored only in the file system where FNAT is located.

When generating DDMs on the mainframe and the FDIC file is also located on mainframe, the DDM will not be available on the Windows or UNIX platform.
The function must be executed twice.

Calling the Function

The Generate a DDM screen is displayed with function code G and object code DD in a Predict main menu, or with the command GENERATE DDM.

```

13:44:32          ***** P R E D I C T  4.3.1  *****                2003-05-31
Plan 10          - Generate a DDM -

Current VM ..... HOME
File ID .....*
Contained in DA .....

Overwrite option ..... Y (Y,N)          For Adabas
List generated code .... Y (Y,N)        Use Vista access-nr ....* N
Adabas version .....* I7                For SQL
                                         Truncate creator ..... N (Y,N)

Field name prefix .....                For IMS
Line comments .....* Y                  Generate UDFs ..... N (Y,N)
Abstract ..... 3 (0-16)                 Replace modified UDFs ... N (Y,N)
General comments ..... Y (Y,N)          List UDFs ..... N (Y,N)
                                         IMS field suffix ..... N (Y,N)

Generate verif. rules .. Y (Y,N)        For Natural Subsystems
Replace verif. rules ..* Y              Generate Security ..... N (Y,N)
List verif. rules ..... Y (Y,N)        Super Natural file opt .. N (Y,N)

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi Selfi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presettings	
The parameters below can be changed in the Modify DDM Defaults screen. See Generation Defaults.	
Specification DB ID	Only applicable to Adabas files and userviews. Y Database ID must be specified. N Database ID may be specified. D Database ID may not be specified This parameter is set to N at installation.
Money format (for SQL)	N Unpacked numeric P Packed numeric

Super Natural file DBNR	The database number of the Super Natural system file must be supplied by the DDA if the Super Natural file is not the current FUSER file at generation time. This parameter is set to 0 at installation.
VSAM trailing BLANK char	If a VSAM prefix name contains trailing blanks, these blanks must be represented in Predict with a special character (because trailing blanks cannot be entered in Predict). When generating a DDM for Natural VSAM the special character specified is removed and the correct representation of a blank inserted. A VSAM trailing BLANK char. is not defined at installation. We recommend setting this parameter to \$.
Super Natural file FNR	The file number of the Super Natural system file must be supplied by the DDA if the Super Natural file is not the current FUSER file at generation time. This parameter is set to 0 at installation.
Parameters	
Current VM	Virtual machine that is currently valid. A read only field. DDMs can be only generated from Predict file objects contained <ul style="list-style-type: none"> • in the database of type local linked via <i>Contains DA</i> to the Current VM or • in databases accessible via the network containing the Current VM. See Generating a DDM for Use with Adabas Vista for a description of how this parameter is evaluated when generating a DDM for use with Adabas Vista.
File ID	ID of the Predict file object from which the definitions are to be generated. Alpha characters must be entered in upper case. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.
Contained in DA	Depending on the setting of Specification DB ID in the Modify DDM defaults screen, entry of a database ID is mandatory, optional, or prohibited for Adabas files and userviews. See Presettings above. If a database is specified, its file list must contain the specified file and the database type must be compatible with the database type specified for this database number within the Natural Parameter Module (via NTDB macro). Enter an asterisk to display a list of databases for selection. Do not enter a Database ID for a DB2 table or view.
Overwrite option	Y An existing DDM is overwritten.
List generated code	Y Generated code is listed.
Adabas version	The version of Adabas for which the DDM is to be generated. Enter an asterisk for valid values or see Adabas Version for more information. The Adabas version you specify for DDM generation is independent of your installation platform.
Field name prefix	Prefix to be used for each generated field name. Note: This parameter does not apply if you are generating a DDM from an SQL file type. See list of SQL file types in Generating DDMs for Use with SQL below.

Line comments	<p>Y The first abstract line from each Predict field object is truncated to 20 characters and appended as a line comment to the generated description.</p> <p>N No line comments are generated from Predict object.</p> <p>O The maximum number of abstract lines defined in Predict (0 - 16) are written as line comments.</p>
Abstract	The number of Predict abstract lines to be included in the generated code as lines beginning with an asterisk.
General comments	<p>Y Comment lines in the Predict file object (marked with ** or */ in field Type) are included.</p> <p>If the field synonym is used, the original field name is displayed. The date and time of the generation and the ID of the user who initiated it are inserted. Source fields of sub/super/hyperdescriptors are also listed.</p>
Generate verif. rules	<p>Y The status of verification objects connected to fields of this file via <i>Is verified by VE</i> is changed to automatic.</p>
Replace verif. rules	<p>Y Verification objects of status automatic connected to fields of this file via <i>Is verified by VE</i> are regenerated if they have been modified since generation.</p> <p>S Verification objects of status automatic connected to fields of this file via <i>Is verified by VE</i> are regenerated only if they are not connected to any other file.</p>
List verif. rules	<p>Y List the IDs of all verification objects connected to each field.</p>
For Adabas	
Use Vista access-nr	<p>See table Meaning of Parameter Use Vista access-nr.</p> <p>Note: This parameter is only applicable to Adabas files and userviews.</p> <p>See also section Adabas Vista in the Predict and Other Systems documentation.</p>
For SQL	
Truncate creator	<p>Y The name of the generated DDM is the original SQL table/view name without the part that identifies the creator.</p> <p>Note: Only applicable to SQL file types. See Generating DDMs for Use with SQL.</p>
For IMS	
Generate UDFs	<p>Y User-defined fields for Natural for DL/I are generated or regenerated. Only applicable to files of type I, J, and K.</p>
Replace UDFs	<p>Y Any user-defined fields are replaced.</p>

List UDFs	Y User-defined fields are to be listed.
IMS field suffix	Y The DDM field name is created using a compressed field name (or Natural synonym), hyphen and IMS segment name.
For Natural subsystems	
Generate security	Only applicable if Natural Security is installed and the user is authorized to create profiles. An authorized user is an administrator who either has no owners or is an owner of himself without countersignatures. Y An authorized user can create a Natural Security profile for the file whose DDM has just been generated.
Super Natural file opt.	This option can only be used by a Super Natural administrator. Y The file is to be declared to Super Natural and causes a list of Super Natural users to be displayed so that the administrator can specify which of them can use the file. If Natural Security is installed, the file is declared to Super Natural only if it is given, or already has, a Natural Security definition.

Generating DDMs for Use with IMS

When generating a DDM for one of the IMS file types, the following fields are included:

For all file types (I, J, K)

- KEY fields from the higher level IMS segments
- KEY fields from the current IMS segment
- KEY fields from the lower level IMS segment if data is accessed via an AIX index

Additionally for IMS Segments (file type I)

- user-defined fields of all IMS segment layouts (file type J)
- user-defined fields of the related logical child or logical parent segment

IMS Segment (file type J)

- user-defined fields of the named IMS segment layout
- user-defined fields of the related logical child or logical parent segment

IMS userview (file type K)

- user-defined fields of the named IMS User View file
- user-defined fields of the related logical child or logical parent segment

Generating DDMs for Use with SQL

Please note the following when generating DDMs for SQL file types:

- Parameter Truncate creator only applies when generating DDMs for the file types listed below.
- Natural synonyms and parameter Field name prefix are ignored for these file types.

SQL file types are

SQL File Types	
A(SQL)	Adabas (with SQL usage set to Y)
AT	Adabas table cluster
B	Adabas SQL view
BT	Adabas D table
BV	Adabas D view
D	DB2 table
E	DB2 view
JT	Ingres table
JV	Ingres view
OT	Oracle table
OV	Oracle view
X	General SQL file
XT	Informix table
XV	Informix view
YT	Sybase table
YV	Sybase view

Generating a DDM for Use with Adabas Vista

Prerequisites

If an Vista access number is used for generating a DDM, the virtual machine of the Vista element used for generation must be linked to the current network.

Changing the Current VM and Current Network is described under Miscellaneous in the **Predict Administration documentation**.

Recommendations when Using Adabas Vista

If using Predict with Adabas Vista distributed data processing, we recommend the following:

- Always enter Vista access numbers for Adabas files and userviews (Add, Modify file).
- Set the parameter Specification DB ID in the Modify DDM defaults screen to D (database ID may not be specified when generating a DDM from an Adabas file/userview) or to N (database ID may be specified but is not mandatory).
If this parameter is set to Y, the user is forced to enter a database ID when generating a DDM from an Adabas file/userview.
- Set the parameter Use Vista access-nr in the Modify DDM defaults screen to T, but do not protect the field.
- Do not enter a database ID with the function Generate a DDM. This ensures the Vista access number is used for generation.

If you do not want to use Adabas Vista:

- Set the parameter Use Vista access-nr in the Modify DDM defaults screen to N and deactivate the option by blanking out the preceding X.

The table below illustrates the use of the parameter Use Vista access-nr.

Meaning of Parameter Use Vista access-nr

Specification DB ID	Use Vista Access-Nr	Database ID specified	Result
D,N	N	no	DDM is generated with DB ID=0
D,N	Y	no	If the file does not contain Vista access numbers, an error message is given. If the file contains Vista access numbers, a DDM is generated with the access numbers taken from the file.
D,N	T	no	If the file does not contain Vista access numbers, an error message is given. If the file contains Vista access numbers, the system tests whether a Vista element with the same numbers exists for this file. If so, a DDM is generated with this access numbers.
N,Y	N	yes	[1] If the file is not defined as partitioned in this database and the Adabas attribute Vista access only is not set to Y: the DDM is generated with the physical database number and physical file number (defined with file maintenance function Modify Adabas attributes).
N,Y	N	yes	[2] If the file is defined as partitioned in this database or the Adabas attribute Vista access only=Y: the system checks whether a Vista element for this file and database exists. If exactly one element exists, the DDM is generated with the Vista numbers defined in this Vista element. If more than one Vista element is found, no DDM is generated and an error message is given.
N,Y	T,Y	yes	Same as above, except that if several Vista elements exist for the file and database, the system checks whether the Vista access numbers of the file are identical to the Vista numbers of one of these Vista elements. If identical, the DDM is generated. If not, no DDM is generated and an error message is given.

Generate DDM in Batch Mode

Command: GENERATE DDM

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Database ID	DATABASE-ID	2
Overwrite option	REPLACE	3
Field name prefix	PREFIX	4
Line comments	LINE-COM	5
General comments	GENERAL-COM	6
Abstract	SHORT-COM	7
List generated code	LIST	8
Generate verif. rules	GENERATE-RULE	9
Replace verif. rules	REPLACE-RULE	10
List verif. rules	LIST-RULE	11
Generate UDFs	GENERATE-UDF	12
Replace UDFs	REPLACE-UDF	13
List UDFs	LIST-UDF	14
Adabas version	ADA-VER	15
Truncate creator	TRUNC-CREATOR	16
IMS field suffix	IMS-SUFFIX	17
Use Vista access nr.	USE-ACCESS-NR	18
Generate Security	ADD-SECURITY	19
Status ¹⁾	STATUS	20
DDM Modifiers ¹⁾	MODIFIER1	21
DDM Modifiers ¹⁾	MODIFIER1-COUNTER-SIGN	22
DDM Modifiers ¹⁾	MODIFIER2	23
DDM Modifiers ¹⁾	MODIFIER2-COUNTER-SIGN	24
DDM Modifiers ¹⁾	MODIFIER3	25
DDM Modifiers ¹⁾	MODIFIER3-COUNTER-SIGN	26
DDM Modifiers ¹⁾	MODIFIER4	27
DDM Modifiers ¹⁾	MODIFIER4-COUNTER-SIGN	28
DDM Modifiers ¹⁾	MODIFIER5	29
DDM Modifiers ¹⁾	MODIFIER5-COUNTER-SIGN	30
DDM Modifiers ¹⁾	MODIFIER6	31
DDM Modifiers ¹⁾	MODIFIER6-COUNTER-SIGN	32

DDM Modifiers ¹⁾	MODIFIER7	33
DDM Modifiers ¹⁾	MODIFIER7-COUNTER-SIGN	34
DDM Modifiers ¹⁾	MODIFIER8	35
DDM Modifiers ¹⁾	MODIFIER8-COUNTER-SIGN	36
Owner ¹⁾	OWNER1	37
Owner ¹⁾	OWNER1-COUNTER-SIGN	38
Owner ¹⁾	OWNER2	39
Owner ¹⁾	OWNER2-COUNTER-SIGN	40
Owner ¹⁾	OWNER3	41
Owner ¹⁾	OWNER3-COUNTER-SIGN	42
Owner ¹⁾	OWNER4	43
Owner ¹⁾	OWNER4-COUNTER-SIGN	44
Owner ¹⁾	OWNER5	45
Owner ¹⁾	OWNER5-COUNTER-SIGN	46
Owner ¹⁾	OWNER6	47
Owner ¹⁾	OWNER6-COUNTER-SIGN	48
Owner ¹⁾	OWNER7	49
Owner ¹⁾	OWNER7-COUNTER-SIGN	50
Owner ¹⁾	OWNER8	51
Owner ¹⁾	OWNER8-COUNTER-SIGN	52
Security Notes ¹⁾	NOTE1	53
Security Notes ¹⁾	NOTE2	54
Security Notes ¹⁾	NOTE3	55
Security Notes ¹⁾	NOTE4	56
Security Notes ¹⁾	NOTE5	57
Security Notes ¹⁾	NOTE6	58
Security Notes ¹⁾	NOTE7	59
Security Notes ¹⁾	NOTE8	60
Security Notes ¹⁾	NOTE9	61
Security Notes ¹⁾	NOTE10	62

¹⁾ Only available in Natural Security. For further details and possible values see the Natural Security documentation.

Note:

When generating a DDM in batch mode for database 0, DATABASE-ID must be set to blank and at least one other parameter must be specified in order that this parameter is recognized. For example:

```
GENERATE DDM
FILE-ID=FILE1, DATABASE-ID= , LIST=Y
```

or in positional form:

```
GENERATE DDM
FILE1, , , , , LIST=Y
```

Sample Output

```
13:19:05          ***** P R E D I C T 4.3.1 *****          2003-05-31
                        - Generate a DDM -                               Page: 1

          DBID: 180  FNR: 171  DDM: GENERATION-EXAMPLE          DEF.SEQ:

T L DB NAME                                     F LENG S D REMARKS
- - - - -
*      Generation started
*      at 2003-05-31 13:19:05
*      by user MSZ
*
*
*      EXAMPLE FILE FOR THE
*      GENERATION SUBSYSTEM
*
G 1 AA GROUP-1
  2 AB ELE-N-9V5          N 9.5  N
    HD=ELE/HEADER

VER: GEN-CHECK-ZERO

G 2 AC GR-IN-GROUP
  3 AD ELE-B-4          B 4.0  F D

VER: GEN-VER-EXAM

  3 AE ELE-PS-5V2          P 5.2  N

VER: GEN-CHECK-ZERO

*      MU field in group redefined two
*      times
M 3 AF MU-B-4          B 4.0  N
  2 AG ELE-A-42          A 42.0 N

VER: GEN-CHECK-BLANK
VER: GEN-VER-EXAM

  2 AH ELE-F-8          F 8.0  F
  2 AI ELE-B-3          B 3.0  N
*      Superfield and superdescriptor
  1 AJ SB-ELE-A-5          A 5.0  N
*      ----- SOURCE FIELD(S) -----
*      ELE-A-42(10-14)
```

```

13:19:05          ***** P R E D I C T 4.3.1 *****          2003-05-31
                          - Generate a DDM -                          Page: 2

          DBID: 180 FNR: 171 DDM: GENERATION-EXAMPLE          DEF.SEQ:

T L DB NAME          F LENG S D REMARKS
- - - - -
1 S1 SP-DE-A-24          A 24.0 N S
* ----- SOURCE FIELD(S) -----
*     ELE-B-3(1-3)
*     ELE-A-42(20-36)
*     ELE-PS-5V2(1-4)
*
* PE-group with automatic counter
*
P 1 AL PC-OCC-7
2 AM PC-ELE-DE-NS-7V3          N 7.3 N D
* counter for MU-field in PE-group
M 2 AN PC-MC-PS-6V1          P 6.1 N D

VER: GEN-CHECK-ZERO

*
G 2 AO PC-GR
3 AP PC-ELE-I-2          B 2.0 F
Format 'INTEGER' changed to 'BINARY'.
3 AQ PC-PS-20V7          P 20.7 N
2 AR PC-ELE-F-4          F 4.0 F
* Formats: date,time,logical
1 AS ELE-D          D 6.0 N
1 AT ELE-T          T 12.0 N
1 AU ELE-L          L 1.0 N
* Hyperdescriptor with MU field
* and source field of a PE-group
M 1 S2 HQ-DE-I-4          I 4.0 H
* ----- SOURCE FIELD(S) -----
*     PC-ELE-I-2
*     MU-B-4
***** DDM replaced *****
    
```

Generate for DB2

Various types of external objects can be generated for DB2. This section contains:

- Common Parameters for all DB2 Types
- DB2 Database
- DB2 Procedure/Function
- DB2 Table/View
- DB2 Storagegroup
- DB2 Tablespace

For the above object types several common parameters can be set that are described under Common Parameters for all DB2 Types.

Prerequisites

Generation of DB2 objects is subject to DB2 security mechanisms. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation** for more information.

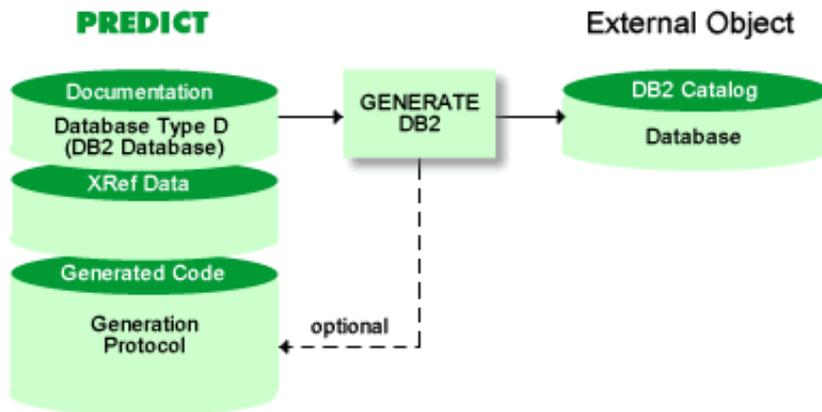
Common Parameters for all DB2 Types

The following parameters can be set in all DB2 object types.

Parameters	
Use SQLID	To switch from the logon user ID to a group ID in DB2, you can enter a different SQL ID in this field. The statement SET CURRENT SQLID will then be submitted.
List DB2 statements	Y List the generated DB2 SQL statements.
Protocol saved in member	SQL statements generated for the implementation of the database can be saved in a Natural source member (protocol) on the FDIC file. A member name for the protocol is only required when the DB2 database is generated for the first time. SQL statements generated in subsequent generation runs are concatenated to the protocol. The protocol can be displayed using the Display function of the Administration Implemented Database menu.
Protocol saved in library	Library where the protocol is to be saved.

DB2 Database

A DB2 database can be implemented with Predict by generating the necessary SQL statements. An additional confirmation is requested before the DB2 database is actually implemented. The generated SQL statements can be saved in a generation protocol.



Note:

This function is only available if DB2 is installed in your environment. The function is not applicable to SQL/DS.

Calling the Function

The Generate DB2 Database screen is displayed with function code G and object code D2 in a Predict main menu, or with the command GENERATE DB2-DATABASE.

Prerequisites

Generation of DB2 objects is subject to DB2 security mechanisms. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation** for more information.

```

13:13:38          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Generate DB2 Database -

Database ID .....*

Add database ..... Y (Y,N)
Update database ..... N (Y,N)

Use SQLID ..... HNO
List DB2 statements ..... Y (Y,N)

Protocol saved in member ... *
                in library .. DB2LIB

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last Lnke1 Flip Print Impl AdmFi Selfi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presetting	
The parameter below can be changed in the Modify DB2 Database Defaults screen. See Generation Defaults.	
Count of saved actions	Number (0-99) of generation actions which are stored in the protocol specified under Protocol saved in member/library. If 99 is entered, all generation actions are stored. This is the default setting at installation.
Parameters	
Database ID	ID of the Predict database object from which the DB2 database is to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.
Add database	Y A new DB2 database is created. If the DB2 database already exists, this parameter must be set to N.
Update database	Y If the database is already implemented, the differences of the implemented database and the Database object in Predict are determined and update commands are generated to modify the implemented database according to the Predict object.

Generate DB2 Database in Batch Mode

Command: GENERATE DB2-DATABASE

Enter parameters on next line in positional or keyword form. Database ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

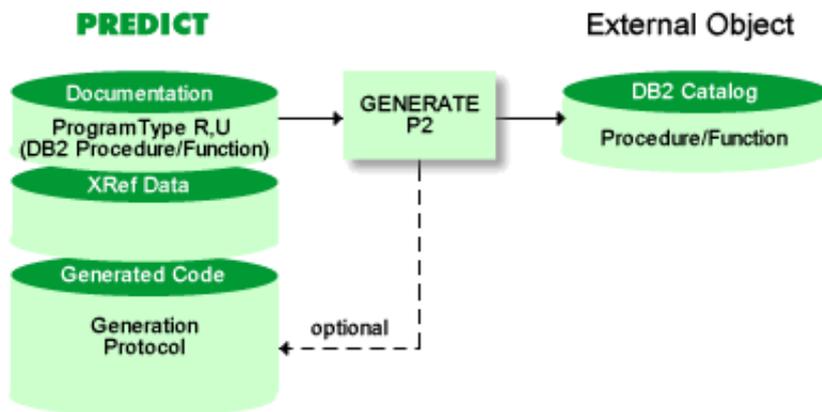
Field	Keyword	Position
Database ID	DB	1
List DB2 statements	LIST	2
Protocol saved in member	MEM	3
Protocol saved in library	LIB	4
Add database	ADD-DB	5
Update database	UPD-DB	6
Use SQLID	SQLID	7

Sample Output

13:51:40	***** P R E D I C T 4.3.1 *****	2003-05-31
	- Generate DB2 Database -	Page: 1
Database-ID: EXAM-DATABASE		
CREATE DATABASE EXAMDB		
STOGROUP EXAM001		
BUFFERPOOL BP1;		

DB2 Procedure/Function

A DB2 procedure or function requires an object of type Program as input, from which then either a procedure or a function is generated.

**Note:**

This function is only available if DB2 is installed in your environment. The function is not applicable to SQL/DS.

Calling the Function

The Generate DB2 Proc/Function screen is displayed with function code G and object code P2 in a Predict main menu, or with the command `GENERATE DB2-PROCEDURE`.

Prerequisites

Generation of DB2 objects is subject to DB2 security mechanisms. See the section `DB2 and SQL/DS` in the **Predict and Other Systems documentation** for more information.

```

13:13:38          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Generate DB2 Proc/Function -

Program ID .....*

Add procedure/function ..... (Y,N)      Replace procedure/function .. (Y,N)
Update procedure/function .. (Y,N)      Comment on ..... (Y,N)

Use SQLID ..... HNO
List DB2 statements ..... Y (Y,N)

Protocol saved in member ... *
                in library .. DB2LIB

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presetting	
The parameter below can be changed in the Modify DB2 Proc/Function Defaults screen. See Generation Defaults.	
Count of saved actions	Number (0-99) of generation actions which are stored in the protocol specified under Protocol saved in member/library. If 99 is entered, all generation actions are stored. This is the default setting at installation.
Parameters	
Program ID	<p>ID of the Predict program object from which the DB2 procedure/function is to be generated.</p> <p>The procedure name or function name is unique in Predict. However, in DB2, there can be several objects with the same name in different schemata. Therefore the object type Program has been extended by the attribute Schema. If no schema is given for a program, the correlation of the documentation and an implementation can be driven by the user, that is:</p> <ul style="list-style-type: none"> • If there is no object with the specified name in DB2, a CREATE statement is generated. • If an object already exists in DB2, you are asked whether the specified object is the correct one, or if you want to create a new object. If the you want to create a new object, a CREATE statement is generated, otherwise an ALTER statement is generated. <p>Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.</p> <p>Note: The option Use SQLID can be set to choose a specific SQLID before the generation. It influences the allocation of the schema to which an object belongs.</p> <p>If no schema was given for an object of type Program, the documentation is supplemented by the SQLID after successful generation.</p>
Add procedure/function	Y A new DB2 procedure/function is created. If the DB2 procedure/function already exists, this parameter must be set to N.
Replace procedure/function	Y An existing procedure/function in DB2 is dropped and created newly.
Update procedure/function	Y If the procedure/function is already implemented, the differences of the implemented procedure/function and the program object in Predict are determined and update commands are generated to modify the implemented procedure/function according to the Predict object.
Comment on	Y The first eight abstract lines of Predict field and file objects are used as DB2 comments for the columns and the table (SQL statement COMMENT ON).

Note:

Links concerning the associations IN (Input FI) and RE (Returns FI) define the input and output parameters of the procedure or function.

Generate DB2 Procedures/Functions in Batch Mode

Command: GENERATE DB2-PROCEDURE

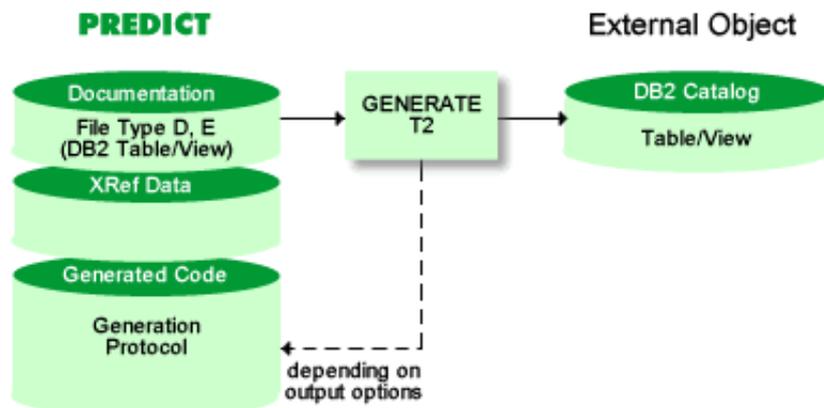
Enter parameters on next line in positional or keyword form. Program ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
Program ID	PROGRAM-ID	1
Add procedure/function	ADD-PROCEDURE	2
Update procedure/function	UPD-PROCEDURE	3
Replace procedure/function	REPLACE	4
Comment on	COMMENT	5
List DB2 statements	LIST	6
Protocol saved in member	MEM	7
Protocol saved in library	LIB	8
Use SQLID	SQLID	9

DB2 Table/View

The Generate DB2 Table / View function:

- Generates DB2 and SQL/DS Table/View/Indexes. If a table contains a LOB column, the function generates a SET CURRENT RULE='STD' statement which allows DB2 to create the necessary auxiliary tablespaces, tables and indexes for these objects.
- Generates distinct types.
- Generates check expressions. A check expression that differs from the documentation in Predict is dropped and recreated.
- Generates triggers. A trigger that differs from the documentation in Predict is dropped and recreated.
- Modifies existing DB2 and SQL/DS Table/Indexes, replaces DB2 and SQL/DS Views/Indexes or
- Purges indexes from DB2 or SQL/DS tables if no longer documented with a corresponding Predict file object of file type D (DB2 Table).



Calling the Function

The Generate DB2 Table/View screen is displayed with function code G and object code T2 in a Predict main menu, or with the command GENERATE TABLE.

Prerequisites

Generation of DB2 objects is subject to DB2 security mechanisms. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation** for more information.

```

13:32:37          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Generate DB2 Table / View -

File ID .....*

Add table / view ..... Y (Y,N)      Replace view ..... N (Y,N)
Add index ..... Y (Y,N)            Defer option for indexes .* Y
Update table ..... N (Y,N)         Label on ..... Y (Y,N)
Update index ..... N (Y,N)         Comment on ..... Y (Y,N)
Purge index ..... N (Y,N)

Use SQLID ..... HNO
List DB2 statements ..... Y (Y,N)

Protocol saved in member ... *
                in library .. DB2LIB

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                Help Next Stop Last LnkJ1 Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presetting	
The parameter below can be changed in the Modify DB2 Table/View Defaults screen. See Generation Defaults.	
Count of saved actions	Number (0-99) of generation actions which are stored in the protocol specified under Protocol saved in member/library. If 99 is entered, all generation actions are stored. This is the default setting at installation.
Parameters	
File ID	ID of the Predict file object from which the DB2 table/view is to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.
Add table/view	Y A new DB2 table/view is created. If the DB2 table already exists, this parameter must be N.

Replace view	<p>Y If the DB2 view already exists, the DB2 statements DROP VIEW and COMMIT are generated first, followed by the CREATE VIEW statement.</p>
Add index	<p>If DB2 indexes are defined for the DB2 table in Predict, set this option to Y to generate the CREATE INDEX statements.</p>
Update table	<p>Y If the DB2 table/view is already implemented, commands are generated to adapt the DB2 table/view to the documentation (if differences were found).</p>
Update index	<p>Y If the DB2 index is already implemented, update commands are generated to adapt the DB2 index to the documentation (if differences were found). Updating DB2 indexes is only possible for differences of the following attributes: buffer pool, close, freepage, pctfree, VSAM catalog, storage group, primary allocation, secondary allocation, erase, index type, GBPCACHE, piece size. If any other attributes differ, Update index and Purge index must be set to Y. Predict will then "update" the index by issuing the statements DROP INDEX, COMMIT and CREATE INDEX.</p>
Purge index	<p>Y A DROP INDEX and a COMMIT statement are generated if a DB2 index exists and no corresponding index is defined in Predict.</p> <p>Note: Partitioned indexes cannot be purged.</p>
Label on	<p>Y The string specified in the Predict field attribute Header1 is used as DB2 label for the columns (SQL statement LABEL ON).</p> <p>Note: If Label on is set to Y, existing labels are generally overwritten even if Header1 is not specified in Predict for each column in the table.</p>
Comment on	<p>Y The first eight abstract lines of Predict field and file objects are used as DB2 comments for the columns and the table (SQL statement COMMENT ON).</p> <p>Note: If Comment on is set to Y, existing comments are generally overwritten even if no abstract is specified in Predict.</p>
Defer option for indexes	<p>Y The SQL statement CREATE INDEX is generated with the option DEFER YES.</p> <p>N The SQL statement CREATE INDEX is generated without the option DEFER YES.</p> <p>S A list containing all indexes is displayed. Enter Y for indexes which you would like to generate with the option. If you enter N or leave the field blank, the CREATE INDEX statement is generated without this option.</p>

Generate DB2 Table/View in Batch Mode

Command: GENERATE TABLE

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Add table/view	ADD-TABLE	2
Add index	ADD-INDEX	3
Update table	UPD-TABLE	4
Update index	UPD-INDEX	5
Purge index	PURGE-INDEX	6
Replace view	REPLACE	7
Note: For compatibility reasons only	PSW	8
List DB2 statements	LIST	9
Protocol saved in member	MEM	10
Protocol saved in library	LIB	11
Label on	LABEL	12
Comment on	COMMENT	13
Defer option for indexes	DEFER	14
Use SQLID	SQLID	15

Sample File Definition

The following file definition was used for generating the sample output:

```

13:21:21          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Display File with Children -                Page:   1

File ID ..... EXAM-GEN_TABLE
-----

DB2 attributes
Number of partitions ..
Edit program ..... PROGDB
Validation program ....
Audit ..... N Audit none
OBid ..... 0
Data capture .....
Max. length ..... 4056

Abstract
this are comments for table
exam-gen_table

Cnt  Ty L Field ID          F  Cs  Length D DB N Df
-----
  1   1 EMPNO              A    6.0  P AA R N
      Field is defined as unique
      Source field(s)          Order
      EMPNO                   ascending
  2   1 FIRSTNME          AV   12.0  N AB R N
  3   1 MIDINIT           A    1.0  N AC R N
  4   1 LASTNAME          AV   15.0  N AD R N
  5   1 WORKDEPT          A    3.0  F AE R Y
      Source field(s)          Order
      WORKDEPT                ascending
  6   1 PHONENO           A    4.0  N AF R Y
  7   1 HIREDATE           D      N AG R Y
  8   1 JOB                A    8.0  N AH R Y
  9   1 EDLEVEL           I    2.0  N AI R Y
 10   1 SEX                A    1.0  N AJ R Y
 11   1 BIRTHDATE          D      N AK R Y
 12   1 SALARY             PS   7.2  N AL R Y
 13   1 BONUS              PS   7.2  N AM R Y
 14   1 COMM               PS   7.2  N AN R Y
*** End of report ***

```

Sample Output

```

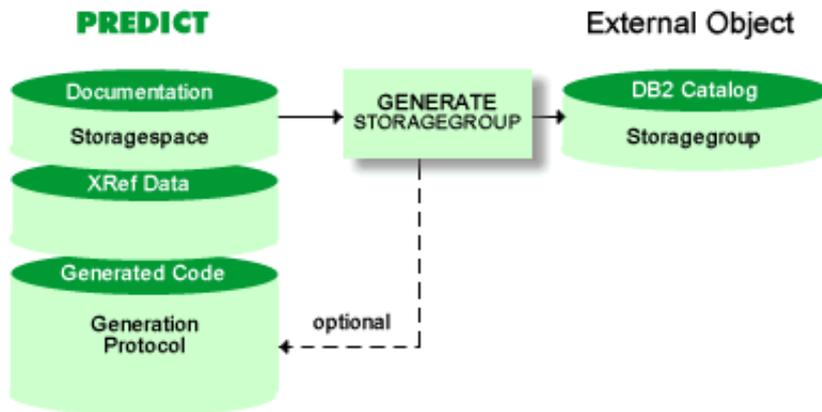
13:52:21          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate DB2 Table / View -                Page: 1
File-ID: EXAM-GEN_TABLE
CREATE TABLE EXAM.GEN_TABLE
  (EMPNO          CHAR(6)          NOT NULL,
   FIRSTNME      VARCHAR(12)     NOT NULL,
   MIDINIT       CHAR(1)          NOT NULL,
   LASTNAME      VARCHAR(15)     NOT NULL,
   WORKDEPT      CHAR(3)          NOT NULL WITH DEFAULT,
   PHONENO       CHAR(4)          NOT NULL WITH DEFAULT,
   HIREDATE      DATE             NOT NULL WITH DEFAULT,
   JOB           CHAR(8)          NOT NULL WITH DEFAULT,
   EDLEVEL       SMALLINT        NOT NULL WITH DEFAULT,
   SEX           CHAR(1)          NOT NULL WITH DEFAULT,
   BIRTHDATE     DATE             NOT NULL WITH DEFAULT,
   SALARY        DECIMAL(9, 2)   NOT NULL WITH DEFAULT,
   BONUS         DECIMAL(9, 2)   NOT NULL WITH DEFAULT,
   COMM         DECIMAL(9, 2)   NOT NULL WITH DEFAULT,
   PRIMARY KEY (EMPNO))
IN DATABASE EXAMDB
EDITPROC PROGDB
AUDIT NONE;
CREATE UNIQUE INDEX PRDDBA.EMPNO
ON EXAM.GEN_TABLE
  (EMPNO          ASC)
SUBPAGES 4
BUFFERPOOL BP0
CLOSE YES;
CREATE INDEX DSN8210.XEMP2
ON EXAM.GEN_TABLE
  (WORKDEPT      ASC)
SUBPAGES 4
BUFFERPOOL BP0
CLOSE YES;

DIC1880 SUMMARY:      14 FIELD(S) PROCESSED

```

DB2 Storagegroup

A DB2 storagegroup can be implemented with Predict by generating the necessary SQL statements. An additional confirmation is requested before the DB2 storagegroup is actually implemented. The generated SQL statements can be saved in a generation protocol.



Calling the Function

The Generate DB2 Storagegroup screen is displayed with function code G and object code SG in a Predict main menu, or with the command GENERATE STORAGEGROUP.

Prerequisites

Generation of DB2 objects is subject to DB2 security mechanisms. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation** for more information.

```

13:09:57          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan              - Generate DB2 Storagegroup -

Storagespace ID .....*

Add storagegroup ..... Y (Y,N)
Update storagegroup ..... N (Y,N)

Use SQLID ..... HNO
List DB2 statements ..... Y (Y,N)

Protocol saved in member ... *
                in library .. DB2LIB

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkeI Flip Print Impl AdmFi SelfI Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presetting	
The parameter below can be changed in the Modify DB2 Storagegroup Defaults screen. See Generation Defaults.	
Count of saved actions	Number (0-99) of generation actions which are stored in the protocol specified under Protocol saved in member/library. If 99 is entered, all generation actions are stored. This is the default setting at installation.
Parameters	
Storagespace ID	ID of the Predict storagespace object from which the DB2 storagegroup is to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.
Add storagegroup	Y A new DB2 storagegroup is created. If the DB2 storagegroup already exists, this parameter must be set to N.
Update storagegroup	Y If the DB2 storagegroup is already implemented, commands are generated to adapt the DB2 storagegroup to the documentation (if differences were found).

Generate DB2 Storagegroup in Batch Mode

Command: GENERATE STORAGEGROUP

Enter parameters on next line in positional or keyword form. Storagespace ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

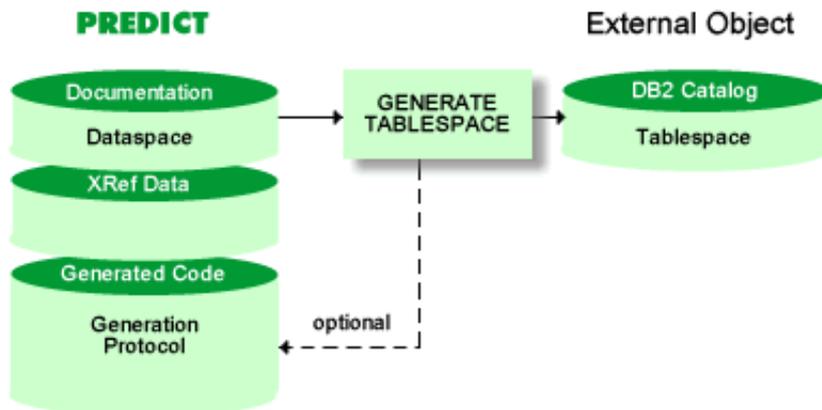
Field	Keyword	Position
Storagespace ID	STORAGESPACE-ID	1
Add Storagegroup	ADD-STORAGEGROUP	2
Update Storagegroup	UPD-STORAGEGROUP	3
List DB2 statements	LIST	4
Protocol saved in member	MEM	5
Protocol saved in library	LIB	6
Note: For compatibility reasons only.	PASSWORD	7
Use SQLID	SQLID	8

Sample Output

13:53:28	***** P R E D I C T 4.3.1 *****	2003-05-31
	- Generate DB2 Storagegroup -	Page: 1
Storagespace-ID: EXAM-STORAGEGROUP		
CREATE STOGROUP EXAM001		
VOLUMES (VOL05)		
VCAT DB2;		

DB2 Tablespace

A DB2 tablespace can be implemented with Predict by generating the necessary SQL statements. Additional confirmation is requested before the DB2 tablespace is actually implemented. The generated SQL statements can be saved in a generation protocol.



Calling the Function

Display the Generate DB2 Tablespace screen with code G and object code TS in a Predict main menu, or with the command GENERATE TABLESPACE.

Prerequisites

Generation of DB2 objects is subject to DB2 security mechanisms. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation** for more information.

```

13:19:18          ***** P R E D I C T  4.3.1  *****                2003-05-31
Plan              - Generate DB2 Tablespace -

Dataspace ID .....*

Add tablespace ..... Y (Y,N)
Update tablespace ..... N (Y,N)

Define ..... Y (Y,N)

Use SQLID ..... HNO
List DB2 statements ..... Y (Y,N)

Protocol saved in member ..... *
                    in library .... DB2LIB

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelfI Prof Main
  
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presetting	
The parameter below can be changed in the Modify DB2 Tablespace Defaults screen. See Generation Defaults.	
Count of saved actions	Number (0-99) of generation actions which are stored in the protocol specified under Protocol saved in member/library. If 99 is entered, all generation actions are stored. This is the default setting at installation.
Parameters	
Dataspace ID	ID of the Predict dataspace object from which the DB2 tablespace is to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.
Add tablespace	Y A new DB2 tablespace is created. If the DB2 tablespace already exists, this parameter must be set to N.
Update tablespace	Y If the tablespace is already implemented, the differences of the implemented tablespace and the documented dataspace are determined and update commands are generated to modify the implemented tablespace according to the documented dataspace. Note: Certain parameters may be modified with the ALTER TABLESPACE statement only when the database is active, other parameters only when the database is stopped. For this reason it may be necessary to submit the generation job twice.
Define	N A DEFINE NO clause is generated

Generate DB2 Tablespace in Batch Mode

Command: GENERATE TABLESPACE.

Enter parameters on next line in positional or keyword form. Dataspace ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
Dataspace ID	DATASPACE-ID	1
Add tablespace	ADD-TABLESPACE	2
Update tablespace	UPD-TABLESPACE	3
List DB2 statements	LIST	4
Protocol saved in member	MEM	5
Protocol saved in library	LIB	6
Note: For compatibilty reasons only.	PASSWORD	7
Define	DEFINE	8
Use SQLID	SQLID	9

Sample Output

```
13:54:53          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate DB2 Tablespace -                    Page: 1
Dataspaces-ID: EXAM-TABLESPACE

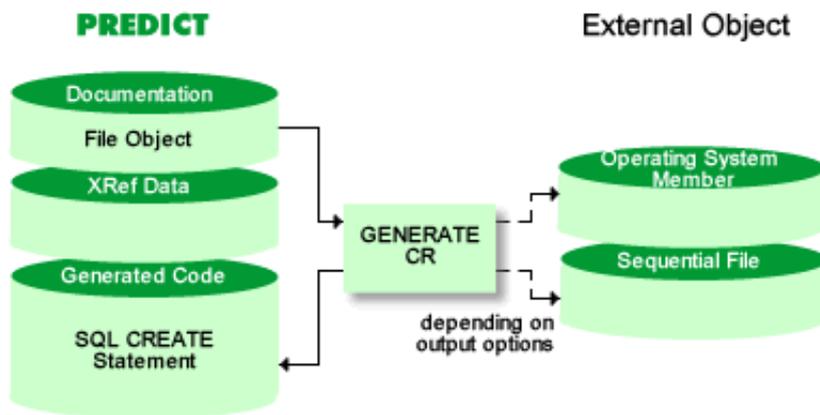
CREATE TABLESPACE EXAMTS IN EXAMDB
  NUMPARTS 4
  BUFFERPOOL BP0
  LOCKSIZE ANY
  CLOSE NO;
```

Generating for SQL

The following topics provide necessary knowledge on Generating for SQL.

SQL CREATE Statements

With the function Generate SQL Create Statements you can generate CREATE TABLE or CREATE VIEW statements. For Adabas SQL Server you can generate CREATE TABLE DESCRIPTION and CREATE CLUSTER DESCRIPTIONS. These SQL statements are stored as Natural members.



Calling the Function

The Generate SQL Create Statements screen is called with function code G and object code CR in a Predict main menu or with the command GENERATE CR.

```

10:41:04          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan 10          - Generate SQL Create Statements -

File ID .....*

Save as member .....          Save in library ..... CRELIB
Overwrite option ..... Y (Y,N) Op. system member .....
Punch / output .....* N

List generated code ..... Y (Y,N)

Truncate creator ..... N (Y,N) Label on ..... Y (Y,N)
Generate defaults ..... Y (Y,N) Comment on ..... Y (Y,N)
Generate procedures ..... Y (Y,N)
Generate indexes ..... Y (Y,N) Defer option for indexes .* N
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Parameters

<p>The parameters Save as member, Save in library, Overwrite option and Op. system member, Punch/output and List generated code are described in the section Generation Defaults.</p>	
File ID	<p>ID of the Predict file object from which the CREATE statement is to be generated.</p> <p>Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.</p> <p>CREATE statements can be generated from the following file types:</p> <p>A, B (with parameter Adabas SQL usage set to Y) Adabas tables and views</p> <p>AT Adabas table clusters</p> <p>D, E DB2 tables and views</p> <p>BT, BV Adabas D tables and views</p> <p>JT, JV Ingres tables and views</p> <p>OT, OV Oracle tables and views</p> <p>XT, XT Informix tables and views</p> <p>YT, YV Sybase tables and views</p> <p>X General SQL file</p>
Truncate creator	<p>Y The creator is truncated from all object names in the CREATE statement.</p> <p>For DB2 CREATE statements: If parameter Generate indexes is set to Y, the creator is truncated in the generated distinct types and indexes, too.</p>
Generate indexes	<p>Y Only applicable to DB2: CREATE INDEX statements are also generated.</p>

Generate defaults	<p>Y For Sybase tables: If default values have been specified in Predict for fields in the file, the following statements are generated:</p> <ul style="list-style-type: none"> ● create default ● sp_bindefault <p>For other SQL systems: A corresponding DEFAULT clause is generated as part of the generated CREATE TABLE statement. If a default for non-null values has been specified in Predict for fields in the file, this value is used in the generated CREATE statement.</p>
Generate procedures	<p>Only applicable to Sybase, Informix and Ingres tables if triggers are defined and linked to the corresponding file object in Predict via <i>Has TR</i> or to field of the file via <i>Triggered by TR</i> and these triggers contain references to procedures.</p> <p>Y For Sybase, Informix and Ingres tables, the statement CREATE PROCEDURE is also generated.</p>
Label on	<p>Y The string specified in the Predict field attribute Header1 is used as label for the columns (SQL statement LABEL ON).</p>
Comment on	<p>Y The first eight abstract lines of the field and file objects in Predict are entered as comments for the columns and the table (SQL Statement COMMENT ON).</p>
Defer option for indexes	<p>Y The SQL statement CREATE INDEX is generated with the clause DEFER YES.</p> <p>N The SQL statement CREATE INDEX is generated without the clause DEFER YES.</p> <p>S A list containing all indexes is displayed. Enter Y for indexes which you would like to generate with the DEFER YES clause.</p> <p>If you enter N or leave the field blank, the CREATE INDEX statement is generated without this clause.</p>

Generating SQL CREATE Statements in Batch Mode

Command: GENERATE SQL-CREATE

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Save as member	MEM	2
Save in library	LIB	3
Overwrite option	REP-OPT	4
Op. system member	OS-MEMBER	5
Punch / output	PUNCH	6
Truncate creator	TRUNC-CREATOR	7
Generate defaults	DEFAULTS	8
Generate procedures	PROCEDURES	9
Generate indexes	INDEX	10
Label on	LABEL	11
Comment on	COMMENT	12
Defer option for indexes	DEFER	13
If you are working with Entire System Server		
- DB-ID	NP-DBID	14
- Dataset	NP-DSNAME	15
- Volume	NP-VOLSER	16
- Library	NP-LIB	17
- Sublibrary	NP-SUBLIB	18
- Member type	NP-MEMTYPE	19
- VSAM catalog	NP-VSAMCAT	20

Sample Output

```

13:50:03          ***** P R E D I C T 4.3.1 *****          2003-05-31
                  - Generate SQL Create Statements -          Page: 1

File ID .. DSN8230-EMPPROJACT

CREATE TABLE DSN8230.EMPPROJACT
  (EMPNO              CHAR(6)          FOR SBCS DATA
                                NOT NULL,
   PROJNO            CHAR(6)          FOR SBCS DATA
                                NOT NULL,
   ACTNO             SMALLINT          NOT NULL,
   EMPTIME           DECIMAL(5, 2),
   EMSTDATE          DATE,
   EMENDATE          DATE,
   UNIQUE (          PROJNO,
                    ACTNO,
                    EMSTDATE,
                    EMPNO),
   FOREIGN KEY REPAE (EMPNO)
     REFERENCES DSN8230.EMP ON DELETE RESTRICT,
   FOREIGN KEY REPAPA (PROJNO,
                     ACTNO,
                     EMSTDATE)
     REFERENCES DSN8230.PROJACT ON DELETE RESTRICT)
IN DATABASE DSN8D23A
AUDIT NONE
DATA CAPTURE NONE;
CREATE INDEX DSN8230.XEMPPROJACT2
ON DSN8230.EMPPROJACT
  (EMPNO              ASC)
  USING STOGROUP DSN8G230
  PRIQTY 12
  SECQTY 12
  ERASE NO
  PCTFREE 10
  SUBPAGES 8
  BUFFERPOOL BP0
  CLOSE NO;
CREATE UNIQUE INDEX DSN8230.XEMPPROJACT1
ON DSN8230.EMPPROJACT
  (PROJNO              ASC,
   ACTNO               ASC,
   EMSTDATE            ASC,
   EMPNO               ASC)

  USING STOGROUP DSN8G230
  PRIQTY 12
  SECQTY 12
  ERASE NO
  PCTFREE 10
  SUBPAGES 8
  BUFFERPOOL BP0
  CLOSE NO;

DIC1800 SUMMARY:    16 FIELD(S) PROCESSED

```

Generating for OS/400

The following topics provide information needed to generate a native OS/400 file:

- Principles
- Defining an OS/400 File Object
- Calling the Generate Function
- Changing an OS/400 Database File

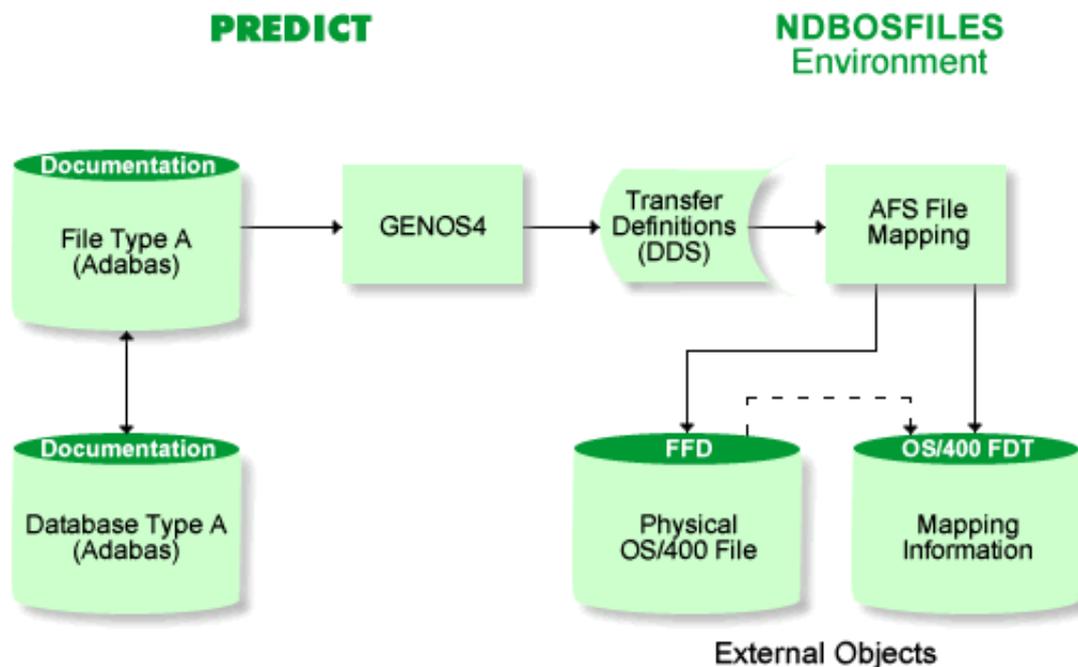
Principles

This section describes the principles of how the Predict file and field view is mapped on OS/400 database files. It also demonstrates the components that are used to create and manage a physical OS/400 file. The following topics are covered:

- Components
- Mapping Rules

Components

The diagram below shows the components in the Predict and AFS File-Mapping environment that enable you to create a physical OS/400 file from a Predict file object.



The following abbreviations are used on the diagram above:

- **PRD**: Product short code for Predict. In this document, PRD also denotes the Natural terminology of file field-level structures.
- **AFS**: The Software AG term AFS is the product short code for the OS/400 Shell Environment that includes all database- and file-mapping programs.
- **FDT**: The Software AG term *Field Description Table* describes the field-mapping information that is stored in the data-mapping library AFSvrsDTA. vrs denotes version, release and SM level.
- **DDS**: The IBM term *Data Description Specification* is a source file that is used to build a new database file.
- **FFD**: The IBM term *File Field Description* describes the field structure of an existing OS/400 database file.

When Natural accesses an OS/400 database file, it uses the FDT information to map its data view to the OS/400 data records. For details, see **Using the AFS File-Mapping Environment** in the Natural for OS/400 Administration document.

Mapping Rules

This section describes the rules of how Predict field types and field formats are mapped to the corresponding OS/400 file field structures.

In principle, Predict assumes that the generated file would be an Adabas file. Because an OS/400 FFD cannot provide all elements used by Adabas, this section also lists the limitations that are caused by differences in the database management systems.

Field Type Mapping

The following table lists how Predict field types (column Ty) are mapped to the corresponding FDT (column T) and OS/400 DDS field attributes:

PRD Field Type	Description	FDT Type in AFS	DDS Source Lines will be:	Remarks
blank	plain field	blank	generated, but not as part of a PE array	
GR	Group	G	not generated	
MU, MC	Multiple value field	M	generated	Overall field length in DDS/FFD = (field length * Occ) + 2
PE, PC	Periodic group	P	generated	Overall field length in DDS/FFD = (sum of all field lengths * Occ) + 2
SB	Subfield	S	not generated	The FDT-field name matches the parent field name. The FDT-field offset is derived from the relative start- and end-character offset within the PRD field.
SP	Superfield	T	indicates the Superfield name	The FDT offsets of the Superfield elements are derived from the relative start- and end-character offsets of the parent fields.
		U	indicates the Superfield elements	

Field Attribute Limitations

Due to the differences between the Adabas data view and the OS/400 file field model, you have to consider the following limitations:

- Descriptors are not allowed for fields within multiple occurrence structures, such as MUs and PEs.
- Redefinitions in the Predict file document are ignored during the generation function.
- Under OS/400 it is efficient if you specify the maximum number of occurrences for field arrays (multiple fields and periodic groups). If a Predict array object does **not** contain a value for occurrence (Column Occ), the generation function creates 199 MU or PE elements (the maximum).
- Multi-occurrence structures are not allowed for parent fields of Sub- and Superfields.
- Parent fields with format "P" are invalid in Super- and Subfields.
- Parent fields with format "N" are invalid in Subfields.
- Predict suppression options are not fully covered under OS/400. See the details:

PRD Suppression Option		FDT Option "S"	Remarks
N	Null suppression	N	
F	Fixed	blank	
blank	Normal suppression	blank	
U	Null allowed		Not allowed in FDT. Generation stopped with Error message displayed.
R	Not Null		Not allowed in FDT. Generation stopped with Error message displayed.

Field Format Mapping

The following table lists various Predict field formats and how they are mapped to the corresponding FDT and OS/400 DDS field formats:

PRD Format	Description	FDT Format	DDS Format	Max. length	Remarks
A	Alphanumeric / Character	A	A	253 bytes	
AV	Varchar	-	-		not processed
B	Binary	B	H	126 digits	
I	Integer				Treated like Format B
N, NS, U, US	Numeric unpacked	N	S	31 digits	
P, PS (in digits)	Packed numeric	P (in bytes)	P (in digits)	16 digits	See the table below for examples.
D	Date	P4 (4 bytes)	P6 (6 digits)		Change message displayed
T	Time	P7 (7 bytes)	P12 (12 digits)		Change message displayed
F	Floating point	-	-		not processed
L	Logical	-	-		not processed

Examples of Packed Format notations:

Format in	Units are in	Ex. 1	Ex. 2	Ex. 3	Ex. 4	Ex. 5	Ex. 6
PRD (Natural)	digits	5.0	6.0	5.2	6.2	7.2	8.2
DDS (OS/400 source)	digits	5P	6P	7P02	8P02	9P02	10P02
FDT (AFS file-mapping)	bytes	3	4	4	5	5	6
FFD (OS/400 buffer length in physical record)	bytes	3	4	4	5	5	6

Defining an OS/400 File Object

▶ To create an OS/400 file object, perform the following steps in Predict:

1. Invoke Predict and add a new database object or select an existing one.

The database object must comply with the following criteria:

- The Predict database object must be of type "A" (Adabas).
- The database number must exist in the AFS Database-Mapping environment with type "blank" (not "ADA" !!).
- The first 10 characters of the "Abstract" field must denote the physical library name of the database. It must exist in the AFS File-Mapping environment. The name will be rejected during the generation function if it exceeds 10 characters or if it contains characters that are invalid within an OS/400 library name.

The following screen excerpt displays a sample database-object definition:

```

13:13:29          ***** P R E D I C T  4.3.1  *****          2003-05-31
                    - Display Database -

Database ID ..... NAT-315-DB
Type ..... Adabas, Local          Added 2002-07-28 at 13:58 by SAG
Physical DBnr ..... 1              Modified 2002-07-29 at 13:31 by SAG
-----

Adabas attributes          Nattural file numbers
Maximal files .....      System file (FNAT) ...
Checkpoint file .....    NAT-Security (FSEC) ..
Adabas security .....    Predict (FDIC) .....
Size of RABN ..... 0
Distr. transaction ... N No
Vista access only .... N

Abstract
  NAT315DB

*** End of report ***
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
                    Quit      RFind Flip -      +      Left Right
    
```

The example shows the specifications of the sample database NAT315DB that is delivered with Natural for OS/400 Version 3.1.5.

2. Add a new file object that must comply with the following criteria:

- The first 10 characters of the "Abstract" field must denote the physical name of the target file. The name will be rejected during the generation function if it exceeds 10 characters or if it contains characters that are invalid within an OS/400 file name.

The following screen excerpt displays a sample file-object definition:

```

13:01:06          ***** P R E D I C T  4.3.1  *****          2003-05-31
                    - Display File -

File ID ..... CUSTOMER
Type ..... Adabas file
File number ..... 19
                                     Added 2002-07-29 at 13:00 by SAG
                                     Modified
                                     Fields modified 2002-07-29 at 13:01 by SAG
-----

File attributes
  Sequence field .....
  Adabas SQL usage ..... N

Abstract
  CUST

Description
  =====
  File additional description

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
          Quit      RFind Flip -      +      Left Right
    
```

where 19 is the Predict file-object number, CUSTOMER the Predict file ID and CUST the name of the physical OS/400 target file.

3. Add fields to your file object. The following screen excerpt displays sample elements that are also used later in this document:

```

>> + Fi: CUSTOMER                               L: 1      S: 9
      Ty L Field ID                               F Cs Length  Occ  D U DB S   All
      * - - - - - * - * - - - - - * - - - - *
          1 CUST-NAME                             A          20.0   D  AA
          1 CUST-NO                               N           8.0   D  AB
      GR 1 ADDRESS                                A          20.0   D  AD
          2 CITY                                  A          20.0   D  AD
          2 STREET                               A          20.0   D  AE
          1 SEX                                   A           1.0   D  AF
          1 DEBIT                                 P           6.2   D  AG
      MC 1 PRODUCTS                              A          10.0   6  AH
      SB 1 CITY-SHORT                             A           3.0   D  S1
    
```

4. Link the new file object to the Adabas database object defined in Step 1:

- You can use the Predict command LINK DATABASE FI <database ID> to invoke the child list of your database.
- Add the file object to the list. In the column **PFnr** specify the file number of the physical target file.
- The physical file number must **not** yet exist in the file list of the target database ID.
If you want to upgrade an existing OS/400 file, see the next topic.

The following screen excerpt displays a child list reflecting the previous database and file sample:

```

>> + DA: NAT-315-DB                               L: 1      S: 11
      All Contains FI                               PFnr  T   Fnr  DDM Impl Other
      -----
      CUSTOMER                                     319  A   19
    
```

Note:

The file ID and the physical file name are different in the example above. However, they can be identical. This applies accordingly to the object-file number and the physical file number and to the database ID and the physical library name.

Calling the Generate Function

With the current Predict Version 4.2 you can call the function Generate an OS/400 File only from the Natural command line using the command GENOS4.

With the next Predict version, this function will be provided as an individual External Object type of the Generate function.

▶ To generate a physical OS/400 file and the corresponding OS/400 FDT, perform the following steps:

1. Start a Natural session in which the Workfile Support 1 is enabled.
2. Logon to SYSDIC and execute the command **GENOS4** from the Natural command line or from the NEXT-prompt.
3. On the input screen, specify the File ID of the file object you plan to generate.

The following excerpt displays a sample "Generate OS/400 File Definitions" screen. The file details of the previous example are used:

```

18:47:44          ***** P R E D I C T  4.3.1  *****                2003-05-31
Plan    0          - Generate OS/400 File Definitions -

File ID ..... CUSTOMER
Phys. File number.....
Contained in DA .....
Phys. Database number ..

List generated code .... Y          (Y,N)

Truncation .....* R                  Validate ..... *

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl - - Impl AdmFi Selfi Prof Main
    
```

The screen's input fields are described below:

Input Field	Explanation / Options	
File ID	The file object to be generated from. This is the only mandatory input field.	
Phys. File number	This file number must not yet be available in the AFS Database-Mapping file list of the target DBID. The physical file name (specified in the first 10 characters of the file's "Abstract" field) must not yet exist in the OS/400 database library.	
Contained in DA	The Predict database name. Note that the database library name will be extracted from the first 10 characters of the database's "Abstract" field.	
Phys. Database number	The DBID of the target database library. Note that this DBID must be available in the AFS Database-Mapping Interface and the library (specified in the first 10 characters of the database's "Abstract" field) must exist physically.	
List generated code	The physical OS/400 file name, file number, database name and database number are always displayed. Depending on this parameter, additionally the following will be displayed:	
	if Y (yes)	the generated code.
	if N (no)	only messages and the affected fields.
Truncation	This parameter describes how long field names should be truncated to 10 characters. The following options are possible:	
	R (right)	Trailing characters will be truncated. This is the default.
	L (left)	Leading characters will be truncated.
	M (middle)	Middle characters will be truncated.
Validate	This parameter indicates how characters that are invalid within OS/400 field names should be handled:	
	Blank	Field names stay unchanged. If invalid characters are identified, error messages will be displayed for the relevant fields.
	A replacement character	This character will replace all invalid characters that are found within field names. Valid replacement characters are the letters A through Z, the digits 0 through 9, and the special characters \$, #, @ and underscore(_).
	*	Invalid characters within field names will be truncated. This is the default.

- Press Enter to start the generation. On the succeeding screen prompt, the system displays the complete information that ought to be transferred to the AFS File-Mapping Interface. The first part lists all file and field details for building an OS/400 FDT in the Mapping Interface. The second part includes the temporary DDS specification for creating the corresponding physical file.

The excerpt below shows sample file field details if parameter "List generated code" was set to "Y":

```

VVVV
13:19:04          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate OS/400 File Definitions -          Page: 1

File ID .. CUSTOMER

* FILENAME          = CUST
* FILENUMBER        = 319
* DATABASENAME      = NAT315DB
* DATABASENUMBER    = 1
  T Lv DB O Name          From F Len  S D U M-Len Occ Remarks
  - - - - -
* 01 AA O CUSTNAME        1 A   20  D
* 01 AB O CUSTNO          21 N   8  D
* G 01 AC ADDRESS
* 02 AD O CITY            29 A   20  D
* 02 AE O STREET          49 A   20
* 01 AF O SEX              69 A   1
* 01 AG O DEBIT            70 P   5           8 digits
* M 01 AH O PRODUCTS      75 A   10           62 6
  - - - - -
* S 01 S1 CITY            A           1 3

**

R CUST
  CUSTNAME 00020A B
  CUSTNO   00008S B
  CITY     00020A B
  STREET   00020A B
  SEX      00001A B
  DEBIT    00008P02B
  PRODUCTS 00062H B
    
```

When you page (by pressing ENTER) through the generated transfer list, all fields that caused problems are followed by a message line, which reports a warning, an error or changes that were performed by the system. See the section Field Check Messages for possible messages. If errors occur, the generation process will be interrupted abnormally.

- If the generation process was performed successfully, you will be requested to execute the file creation. A request similar to the one below appears:

```

13:26:09          ***** P R E D I C T  4.3.1  *****                2003-05-31
                    - Generate OS/400 File Definitions -

File ID ..... CUSTOMER                                PFnr ...   319
Database ID .. NAT-315-DB                             PDBnr ..    1

                    +-----+
                    !           Do you want to execute: Y (Y/N)           !
                    +-----+

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnKE1 -      -      Impl AdmFi Selfi Prof Main
    
```

The system will confirm the successful file creation. The following screen excerpt reflects the sample used above:

```

MORE
Page      2                                02-07-31  13:19:04

DIC1800 SUMMARY:      9 FIELD(S) PROCESSED.
    
```

The current step also inserted the new file's FDT into the File-Mapping data of the AFS Database Environment. Before Natural can access the new file, you must perform the steps described in the following paragraph.

▶ **To edit and catalog the generated OS/400 FDT information, perform the following steps:**

1. Invoke the **File Entries and Associations** menu from the OS/400 Database Shell main menu by selecting Function Code **F** and the database ID that you specified as Physical DBnr in Predict. The file you just created should appear in the file list of your database.
2. Issue the command EDT or press PF6-EDT to edit the new file entry. The field list appears. First, press PF4-CHK to allow the check program to complete some information, e.g. adding field lengths to group definitions.

Continuing with the example above, you should get the following field description table (FDT):

13:07:37		SOFTWARE AG - OS/400 Database Shell						7/31/02						
DB-Nbr	1	NAT315DB		File-Nbr	319	CUST	Status UNCATALOGED							
I	T	L	DB	O	Name OS400	From	F	Leng	S	D	U	M-Len	Occ	Remarks
			1	AA	O CUSTNAME	1	A	20		D				
			1	AB	O CUSTNO	21	N	8		D				
	G		1	AC	ADDRESS	29		40						
			2	AD	O CITY	29	A	20		D				
			2	AE	O STREET	49	A	20						
			1	AF	O SX	69	A	1						
			1	AG	O DEBIT	70	P	5						
	M		1	AH	O PRODUCTS	75	B	10				62	6	
	S		1	S1	CITY	29	A	3						
Command _____														
Enter	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12		
			EXIT	CHK	SAV	CAT	UNC							

The FDT is still in UNCATALOGED state. You could now edit/modify field details, like adding descriptors. However, the physical layout of the fields cannot be changed any more. It is good practice to apply all changes also in the corresponding Predict file document to guarantee consistency for later file generations.

3. Issue the command CAT or press PF6-CAT to catalog the file information. During this step, also the logical files (descriptors) will be created in the file's library.

If the cataloging has been ended successfully, the file is ready to be accessed by Natural.

Field Check Messages

Type	Message	Explanation / Action
Error	END-VALUE NOT WITHIN SOURCE FIELD	The end value of a Superfield is not located within a source field.
	SOURCE FIELD HAS INVALID FORMAT	Parent fields with Format "P" are invalid in Super- and Subfields. Parent fields with Format "N" are invalid in Subfields.
	INVALID SOURCE FIELD	Multi-occurrence structures are not allowed for parent fields of Sub- and Superfields.
	RECORD IS LONGER THAN 32K	OS/400 only supports file records shorter than 32K.
	INVALID FIELD NAME	The field name contains invalid characters and the Validate parameter is blank.
	FIELD IN USERVIEW LONGER THAN IN MASTERFILE	A field in a userview cannot be longer than the corresponding field in the master file.
	INVALID SUPPRESSION OPTION	The suppression options "U" and "R" are not allowed for OS/400 files.
	DUPLICATE FIELD-NAME GENERATED	OS/400 database-field names must be unique.
Warning	NUMBER OF OCCURRENCES MISSING	On OS/400, the number of occurrences has been set to the default value of 199.
Change	FORMAT NOT SUPPORTED	The specified field format is not supported on OS/400. It was changed to the format displayed.
	DESCRIPTOR OPTION DELETED	An invalid descriptor option was deleted. Multiple fields and periodic groups mapped to an OS/400 file field cannot contain descriptors.
	FORMAT 'D'/'T' CHANGED TO 'P'	The OS/400 field description table does not support Date or Time. The format was changed to packed.
	UNIQUE OPTION DELETED	The unique option was deleted because the field has no descriptor.
	MULTI BYTE CHARACTER SET SUPPRESSED	Mixed data character set is not supported.
	FIELD NAME TRUNCATED	The field name was shortened to 10 characters corresponding to the truncation rules specified.
	FIELD NAME SET TO FILLER	The field name was set to FILLER <i>n</i> because it consisted of invalid characters only.

Changing an OS/400 Database File

Important: If you plan to change the physical layout of an existing OS/400 database file, your starting point must always be the Predict file definition.

▶ **To change the layout of an OS/400 database file, proceed as follows:**

1. From the OS/400 Database-Shell menu, invoke the Field Entries and Descriptions menu of your file to be changed. Uncatalog the file and remove it from the file list. The file is now no longer available for the mapping interface. In addition, all logical files (descriptors) in the file's library have disappeared.
2. Rename the physical OS/400 file to a save name.
3. Invoke Predict and apply your modification to the relevant file object. For example, add fields, delete fields or enhance field lengths.

Warning:

You must not change field names, otherwise the *MAP option of the CPYF copy file command will not find the counter field in the saved file. See the FMTOPT option below.

4. Generate a new OS/400 file from the Predict file object that you modified in the previous step. During this run, the corresponding file and field definitions will be added again to the AFS File-Mapping information. See the section Calling the Generate Function on how to perform the Generate function for OS/400 files.
5. From the OS/400 Database-Shell menu, invoke the Entries and Associations Menu to edit, check and catalog the new file's FDT. The logical files (descriptors) will then be available again in the file's library.
6. Copy the data contents on a field-to-field basis from the saved file (see Step 2) to the new empty file using the OS/400 command CPYF. The Copy File run will consider the changed layout if you specify the option FMTOPT(*MAP). For example, new fields will be padded according the characteristic of the new file's field description. Use additionally the option FMTOPT(*DROP) to drop those fields in the save-file record format for which there are no fields of the same name in the new-file record format. For more details on the FMTOPT parameter, refer to IBM's File Management documentation (scan for the section "database-to-database copies").
7. According to the changes in the file field layout, you may have to adjust the corresponding DDM definition, e.g. by generating the new DDM from the Predict file object.
8. The OS/400 file is now ready to be accessed by Natural.

Generate For Third Generation Languages - 3GL

The following types of external objects can be generated for 3GL:

- Assembler Copy Code
- C Include Code
- Cobol Copy Code
- FORTRAN Copy Code
- PL/I Include Code

Additional Usage Information for Fields in a File

To enable the Predict Preprocessor to create consistent XRef data for a member where the corresponding file has been modified since generation, Predict generation functions for 3GL members append the generated code with the following information:

- a line containing the text 'FIELD USAGE INFORMATION'
- a list of all fields contained in the file together with usage information. Usage is indicated with a preceding C or U (COUNT or UPDATE).

Please note the following for generated members containing this additional usage information:

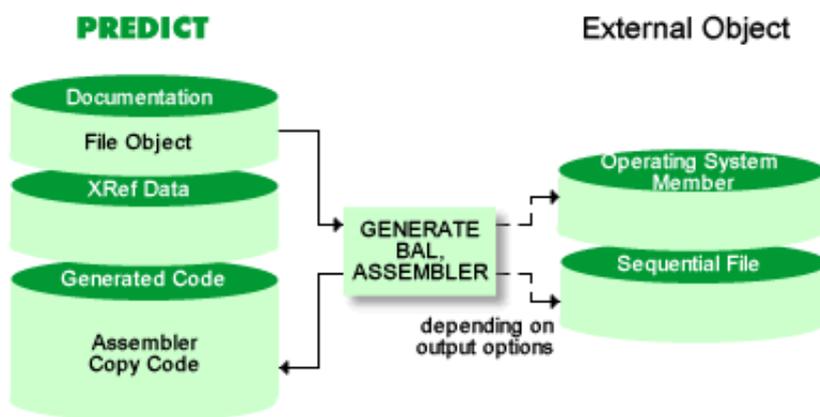
- If you copy a member using the Preprocessor command EXEC Adabas COPY and set the parameter Ignore changes to X, the XRef data is written on the basis of this additional information and not from the current documentation in Predict. This results in consistent XRef data, even if the file has been modified since generation.
- This information is not displayed with Administration functions such as Display implementation. Only the actual member is displayed.
- The same applies to the commands PUNCH/WRITE and to PUNCH with Entire System Server: only the actual member is written

Assembler Copy Code

The function Generate Assembler Copy Code generates a record buffer for use in Assembler programs based on a Predict file object.

In addition, an Adabas format buffer can also be generated if required for files of the following types:

- Adabas file (file type A) with parameter Adabas SQL usage = N
- Adabas userview (file type U)



This section contains:

- Calling the Function
- Generate Assembler Copy Code in Batch Mode
- Names in Assembler Copy Code
- Field Format and Assembler Copy Code
- Sample Output

Calling the Function

The Generate Assembler Copy Code screen is displayed with function code G and object code BA in a Predict main menu or with the command GENERATE BAL or GENERATE ASSEMBLER.

```

09:43:39          ***** P R E D I C T  4.3.1  *****                2003-05-31
Plan    0          - Generate ASSEMBLER Copy Code -

File ID .....*

Save as member .....                               Save in library .... BALLIB
Overwrite option ..... Y (Y,N)                    Op. system member ..
Punch / output .....* N

List generated code ..... Y (Y,N)                  List offsets .....* N
Generate format buffer .....* N                    Adabas version .....* I7

As DSECT ..... N (Y,N)                             Field name prefix ..
With DC or DS .....* DC                            Field name suffix ..
Nr. of abstract lines ..... 4 (0-16)                Validate ..... *
Generate initial value ..... N (Y,N)                Truncation .....* R
Align .....* N                                       With EQUs ..... Y (Y,N)
DSECT name .....                                    Counter length ..... 1 (1,2)
Format buffer name .....

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presettings	
The parameter below can be changed in the Modify BAL Defaults screen. See Generation Defaults.	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated. See Parameters Specifying the Form of Output for more information.
Parameters	
The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output, Workfile name and List generated code are described in the section Parameters Specifying the Form of Output.	
File ID	ID of the Predict file object from which the definitions are to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.

List offsets	<p>Y Include the offset of each item in the record buffer structure (relative to the beginning of the structure) in decimal and hexadecimal formats as a comment. The total length of each buffer is also included.</p> <p>P Include the absolute position (offset+1) as a comment.</p> <p>L Include the total lengths of the record buffer and the format buffer as a comment.</p> <p>V Only allowed if parameters As DSECT=N and With DC or DS=DC. The file number and the calculated lengths of the record buffer and the format buffer are to be generated as constants in the copy code. The name of the file number constant is the record buffer name with N as prefix. The name of each length constant is the appropriate buffer name with L as prefix. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.</p> <p>N No offset.</p>
Generate format buffer	<p>Format buffer generation for Assembler copy code is only allowed if parameters As DSECT=N and With DC or DS=DC. The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas SQL usage set to N) or for files of type U.</p> <p>Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, set this parameter to F or N.</p>
Adabas version	<p>The version of Adabas for which the copy code is to be generated. Enter an asterisk for valid values or see table in the section Adabas Version.</p>

As DSECT	<p>The effect of this parameter depends on the parameter DSECT name. See table below</p> <table border="1" data-bbox="354 353 1209 770"> <thead> <tr> <th data-bbox="354 353 496 398">As DSECT</th> <th data-bbox="496 353 671 398">DSECT name</th> <th data-bbox="671 353 1209 398">Copy Code Generated</th> </tr> </thead> <tbody> <tr> <td data-bbox="354 398 496 454">Y</td> <td data-bbox="496 398 671 454">specified</td> <td data-bbox="671 398 1209 454"><DSECT name> DSECT</td> </tr> <tr> <td data-bbox="354 454 496 510">Y</td> <td data-bbox="496 454 671 510">blank</td> <td data-bbox="671 454 1209 510"><File-ID> DSECT</td> </tr> <tr> <td data-bbox="354 510 496 611">N</td> <td data-bbox="496 510 671 611">specified</td> <td data-bbox="671 510 1209 611"> <DSECT name> DS OCLnnn Note.: nnn=Length of the record buffer structure </td> </tr> <tr> <td data-bbox="354 611 496 667">N</td> <td data-bbox="496 611 671 667">blank</td> <td data-bbox="671 611 1209 667">Only entries for fields are generated</td> </tr> <tr> <td data-bbox="354 667 496 770">N</td> <td data-bbox="496 667 671 770">* (asterisk)</td> <td data-bbox="671 667 1209 770"> <File-ID> DS OCLnnn Note.: nnn=Length of the record buffer structure </td> </tr> </tbody> </table>	As DSECT	DSECT name	Copy Code Generated	Y	specified	<DSECT name> DSECT	Y	blank	<File-ID> DSECT	N	specified	<DSECT name> DS OCLnnn Note.: nnn=Length of the record buffer structure	N	blank	Only entries for fields are generated	N	* (asterisk)	<File-ID> DS OCLnnn Note.: nnn=Length of the record buffer structure
As DSECT	DSECT name	Copy Code Generated																	
Y	specified	<DSECT name> DSECT																	
Y	blank	<File-ID> DSECT																	
N	specified	<DSECT name> DS OCLnnn Note.: nnn=Length of the record buffer structure																	
N	blank	Only entries for fields are generated																	
N	* (asterisk)	<File-ID> DS OCLnnn Note.: nnn=Length of the record buffer structure																	
Field name prefix	The prefix to be used for each field name generated.																		
With DC or DS	<p>DC Assembler DC (define constant) instruction is to be used. If AS DSECT=Y, this parameter must be set to DS.</p> <p>DS Assembler DS (define storage) instruction is to be used.</p>																		
Field name suffix	The suffix to be used for each field name generated.																		
Nr. of abstract lines	The number of Predict abstract lines per field to be included in the generated code.																		
Validate	<p>Determines how invalid characters are handled.</p> <p>blank Invalid characters in a field name will result in an error message but will not be modified.</p> <p>rep.char Invalid characters in a field name are replaced by this character. Valid replace characters: letters A-Z, digits 0-9, \$, % and #.</p> <p>* Invalid characters in a field name are deleted.</p>																		
Generate initial value	<p>This option takes effect only when With DC or DS=DC and As DSECT is set to N.</p> <p>N No initialization.</p> <p>Y Statements are generated to initialize the structure with the value specified for Init value in the corresponding field object in Predict. Fields with no value for Init value are initialized with zeros or blanks. Aligned 8-digit fields with format B or I are not initialized. In a PE group with the 3GL specification Gr.structur set to blank, only the first occurrence of each field is initialized.</p>																		

Truncation	<p>Specifies which characters are deleted if a generated field name is too long:</p> <p>L truncate from the left</p> <p>R truncate from the right</p> <p>M truncate from the middle</p> <p>A warning is given if field names are truncated.</p>
Align	<p>Determines which fields are to be aligned. Fields are aligned on the boundary shown in the table in the section Field Format and Assembler Copy Code.</p> <p>Y All appropriate fields are aligned.</p> <p>N No fields are aligned.</p> <p>S Fields are aligned only if the corresponding Predict field object has the 3GL specification Synchronized=S.</p>
With EQUs	<p>Y EQU statements are to be generated for fields of length 1 whose format is not P and comment lines are to be generated for other fields, using any condition names defined as attributes of the Predict field objects. These names are prefixed, suffixed, validated and truncated in the same way as field names.</p> <p>For fields of format L where no condition name was specified, a condition name is generated by concatenating the field name "Example" to the prefix N.</p> <p>In this case the following statement is generated: NEXAMPLE EQU X'00'.</p>
DSECT name	<p>Specifies the name of the record buffer in the generated structure. The effect of this parameter depends on parameter AS DSECT. See above in this table.</p>
Counter length	<p>Length of additional counter fields. Valid values: 1, 2</p>
Format buffer name	<p>Specifies the name of the format buffer in the generated structure. If omitted, the file ID prefixed by F is used.</p>

Names in Assembler Copy Code

The following rules apply to the generation of names for Assembler copy code.

- Alpha characters in Predict object IDs, such as file and field IDs, are converted to upper-case.
- Each field name (label) used in Assembler copy code is taken from the corresponding Predict field object: either the field name synonym for Assembler or, if none exists, the Predict field ID.
- All field names and any condition names are prefixed, suffixed, validated and - if necessary - truncated to 8 characters according to the parameter settings.
For example, with Field name prefix set to JUN, Field name suffix to *86, Validate to § and Truncation to R two fields JUNCDAY§ and JUNDAY§8 are generated for a field of type MC called DAY.
- If a resulting field name is not unique within the entire file structure, an error message is given.
- Any DSECT name or format buffer name does not receive the field name prefix and suffix. The value

supplied may not contain invalid characters.

- If the file ID is taken as default DSECT name or, with the prefix F, as default format buffer name, it is validated and if necessary truncated.

Assembler Names for Counter Fields - MC or PC

Before each field of type MC or PC, a counter field is generated automatically in the copy code, with a name created from the name of the MC or PC field prefixed by C.

For example, a Predict field object DAY with no field name synonym for Assembler and with type MC will generate two fields with the names CDAY and DAY.

Assembler Names for Logical Fields - L

If no condition name is defined for a logical field (format L), the field name prefixed by N is used for the EQU statement.

Assembler Names for Fields of Type AV

For fields of format AV, a group is generated where groupname=fieldname. The group contains a length field and a text field prefixed with L and T respectively. See second table in the section Field Format and Assembler Copy Code.

Assembler Names for Additionally Generated Indicator Fields

An additional indicator field prefixed with S is generated in the following cases:

- a field is defined with Suppression option set to U (null allowed)
- a field of a file of type A, U or B is defined with Suppression option set to R (not null)

Note:

For Adabas Fields, the additional indicator field is only generated for the following Adabas versions:

- I3 or above
- U1 or above
- V4 or above.

Field Format and Assembler Copy Code

Fields in the Assembler copy code have a clause determined by the following factors:

- length and format of the corresponding Predict field object
- file type of the file containing the field

Predict Format	File Type	PRD Length	Alignment Boundary	Assembler Clause	(if aligned)	Note
B		1		XL1		
		$l=3,5,6,7$		XLl		
I		1		XL1		
B/I		2	2	XL2	H	
		4	4	XL4	FL4	
		8	4	XL8	FL8	
B		$l \Rightarrow 9$		PIC X(l)		
F		4	4	CL4	E	
		8	8	CL8	D	
N/U NS/US ($nn+m < 17$)		$nn.m$		ZLl		$l=nn+m$
N/U NS/US ($nn+m > 16$)		$nn.m$		CLl		$l=nn+m$
P/PS		$nn.m$		PLl		$l=(nn+m+2)/2$
D	D, E			CL10		
	other			PL4		
T	D, E			CL8		
	other			PL7		
L				XL1		
G		n		CLm		$m=2*n$
GL		n		CLm		$m=2*n$
GV		n		CLm		$m=2*n$
LO	D, E		4	XL4	FL4	

The clause XLn is normally generated for B formats. There are two exceptions to this rule:

- parameter Align is set to Y, or
- parameter Align is set to S and the field is defined in Predict as Synchronized.

The clause H or FL4 or FL8 is normally generated for I formats. The exception to this rule is when the parameter Align is set to N and the field does not start at a "synchronized" address.

An indicator or length field is generated like a B2 field. A counter field is also generated like a B2 field if Counter length = 2. If Counter length = 1, the clause XL1 is generated.

A numerical format field with a length ≥ 17 is treated in Assembler copy code as alphanumeric format field. No INIT VALUE can be generated for these fields and a warning is given.

A DB2 large object is generated as locator like an I4 field.

Note:

When generating Assembler copy code for DB2 tables and views, format NS or US is converted to Assembler format PL. Any redefinitions that exist for this field are skipped.

File Type in Predict	Predict Format	Length	Assembler Clause	Note
D, E	A, AL	n	CLn	
	AV	Two-level group is generated: Fieldname DS CLm FieldnameL Dx yyy FieldnameT Dx CLn		m = n + 2 Dx is either DS or DC, depending on parameter With DC or DS. yyy is either XL2 or H, like a B2 field.
A,U,B	A	n	CLn	
	AV	Two-level group is generated. See above.		

Sample Output

```

13:27:40          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate ASSEMBLER Copy Code -          Page: 1

File ID .. GENERATION-EXAMPLE

*****
*   THIS RECORD-BUFFER LAYOUT WAS GENERATED BY PREDICT
*   FOR FILE: GENERATION-EXAMPLE
*   ON: 2003-05-31 STARTING AT 13:27:40
*   FILE COMMENTS: Example file for the
*                   PREDICT generation subsystem
*
*   ..
*****
GROUP1  DS      0D
ELN9V5  DC      ZL14'0'          ELE-N-9V5
GRINGR  DS      0CL30
ELEB4   DC      FL4'0'          ELE-B-4
*
*                   CONDITION NAMES:
*                   INIT-ELE-B-4=00001000
*                   COND-ELE-B-4-V-1=FFFFFFFF
ELPS52  DC      PL4'0'          ELE-PS-5V2
MUB4    DC      5FL4'0'          MU-B-4
        ORG     MUB4
MURB7   DC      XL7'0'          MU-RED-B-7
MURB13  DC      ZL13'0'          MU-RED-US-13
        ORG     MUB4
MURB8   DC      XL8'0'          MU-RED-B-8
*
*                   CONDITION NAMES:
*                   MU-RED-B-8-COND1=111111

```

```

13:27:40          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate ASSEMBLER Copy Code -          Page: 2

      DS      CL12
ELEA42 DC      CL42' '          ELE-A-42
*
*          CONDITION NAMES:
*          COND-NAME-START=COND-START
*          *THRU=COND-ZEND

ELEF8  DC      D'0'          ELE-F-8
ELEB3  DC      XL3'0'        ELE-B-3
CPCOCC7 DC     XL1'0'        CNT OF PC-OCC-7
PCOCC7 DS      0CL76
PCNS7V3 DC     ZL10'0'      PC-ELE-DE-NS-7V3
CPCMCPS DC     XL1'0'        CNT OF PC-MC-PS-6V1
PCMCPS DC     11PL4'0'      PC-MC-PS-6V1
PCGR   DS      0CL17
BALI2  DC      H'0'          PC-ELE-I-2
PCPS20V7 DC    PL14'0'      PC-PS-20V7
BALF4  DC      E'0'          PC-ELE-F-4
      DS      6CL76
ELED   DC      PL4'0'        ELE-D
ELET   DC      PL7'0'        ELE-T
ELEL   DC      XL1'0'        ELE-L
NELEL  EQU     X'00'

*****
*   THIS FORMAT-BUFFER WAS GENERATED BY PREDICT
*   FOR FILE: GENERATION-EXAMPLE
*   ON: 2003-05-31 STARTING AT 13:27:40
*****
FGENERAT DS     0CL295
* >>> DIC1809 WARNING: FIELDNAME(S) TRUNCATED
      DC      CL50'AB,2X,AD,AE,AF1-5,AG,2X,AH,AI,ALC,AM1,AN1C,2,AN1(1'
      DC      CL50'-11),1X,AP1,AQ1,AR1,AM2,AN2C,2,AN2(1-11),1X,AP2,AQ'
      DC      CL50'2,AR2,AM3,AN3C,2,AN3(1-11),1X,AP3,AQ3,AR3,AM4,AN4C'
      DC      CL50',2,AN4(1-11),1X,AP4,AQ4,AR4,AM5,AN5C,2,AN5(1-11),1'
      DC      CL50'X,AP5,AQ5,AR5,AM6,AN6C,2,AN6(1-11),1X,AP6,AQ6,AR6,'
      DC      CL45'AM7,AN7C,2,AN7(1-11),1X,AP7,AQ7,AR7,AS,AT,AU.'
*****

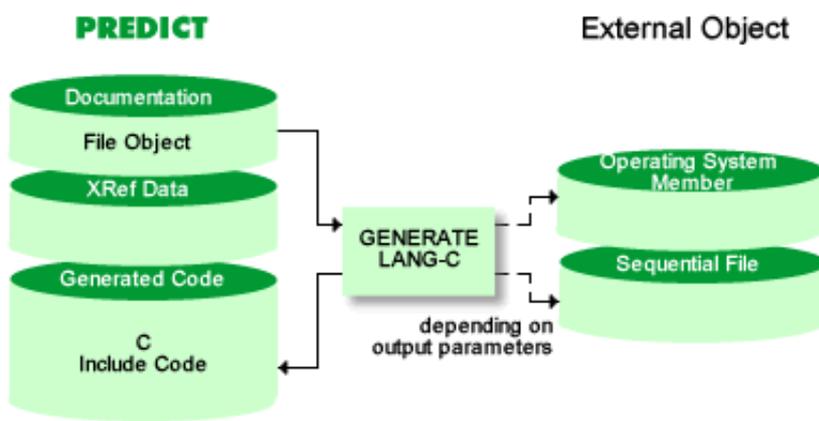
DIC1800 SUMMARY:    25 FIELD(S) PROCESSED
DIC1809 WARNING:    1 FIELDNAME(S) TRUNCATED
DIC1819 MESSAGE:    11 SLACK BYTE(S) GENERATED
DIC1847 MESSAGE:    3 FIELD(S) SKIPPED FOR RECORDBUFFER STRUCTURE
    
```

C Include Code

The function Generate C Include Code generates a record buffer for use in C programs based on a Predict file object.

In addition, an Adabas format buffer can also be generated if required for files of the following types:

- Adabas file (file type A) with parameter Adabas SQL usage = N
- Adabas userview (file type U)



This section contains:

- Calling the Function
- Generate C Include Code in Batch Mode
- Names in C Include Code
- Field Format and C Include Code
- Sample Output

Calling the Function

The Generate C Copy Code screen is displayed with function code G and object code CC in a Predict main menu, or with the command GENERATE LANG-C.

```

10:00:41          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Generate C Copy Code -

File ID .....*

Save as member .....          Save in library .... CCCLIB
Overwrite option ..... Y (Y,N) Op. system member ..
Punch / output .....* N

List generated code ..... Y (Y,N) List offsets .....* N
Generate format buffer .....* N Adabas version .....* I7
Check field name .....* A

Shift increment ..... 3 (0-9) Field name prefix ..
Nr. of abstract lines ..... 3 (0-16) Field name suffix ..
Storage class .....* A Validate ..... _
Upper or lower case .....* L Truncation .....* R
Record buffer name .....
Format buffer name .....

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJ1 Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presettings	
The parameters below can be changed in the second Modify C Defaults screen. See Generation Defaults.	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated. See Parameters Specifying the Form of Output for more information.
Max. name length	Maximum length of name in C.
Signific. length	Number of characters that are used for uniqueness check. May not be greater than Max. name length.
Represent. of I2	Determines how 2-byte integers are represented in the generated code. C unsigned character, 2 bytes S short I int

Represent. of I4	<p>Determines how 4-byte integers are represented in the generated code.</p> <p>C unsigned character, 4 bytes</p> <p>I int</p> <p>L long</p> <p>Note: It is not possible for both I2 and I4 to be represented by int.</p>
Open square bracket	Up to 5 characters used to represent the character [.
Close square bracket	Up to 5 characters used to represent the character].
Open brace	Up to 5 characters used to represent the character {.
Close brace	<p>Up to 5 characters used to represent the character }.</p> <p>Note: All characters are valid apart from the plus sign (+). Characters can be specified in hexadecimal notation. If either the normal or hexadecimal notation for any of these four parameters is changed, the alternative notation must be deleted. Predict then inserts the correct value in either normal or hexadecimal format.</p>
Parameters	
The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output, Workfile name and List generated code are described in the section Parameters Specifying the Form of Output.	
File ID	<p>ID of the Predict file object from which the definitions are to be generated.</p> <p>Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.</p>

List offsets	<p>Y Include the offset of each item in the record buffer structure (relative to the beginning of the structure) in decimal and hexadecimal formats as a comment. The total length of each buffer is also included.</p> <p>P Include the absolute position (offset+1) as a comment.</p> <p>L Include the total lengths of the record buffer and the format buffer as a comment.</p> <p>V The file number and the calculated lengths of the record buffer and the format buffer are to be generated as constants in the include code. The name of the file number constant is the record buffer name prefixed by N_. The name of each length constant is the appropriate buffer name prefixed by L_. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.</p> <p>N No offset.</p>
Generate format buffer	<p>The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas SQL usage set to N) or for files of type U.</p> <p>Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, set this parameter to F or N.</p>
Adabas version	The version of Adabas for which the include code is to be generated. Enter an asterisk for valid values or see table in the section Adabas Version for more information.
Check field name	<p>A Type names are checked for uniqueness (against other type names) within the whole structure. Field names must be unique as field names within the entire structure.</p> <p>Y As above, but duplicate field names are only identified as errors if this would result in compiler errors.</p> <p>N No check for duplicate names is performed.</p>
Shift increment	The number of positions to be shifted right when an open brace (or substitute character string, see Presetting above) is encountered.
Field name prefix	The prefix to be used for each name generated.

Nr. of abstract lines	Number of Predict abstract lines per field to be included in the generated code.
Field name suffix	The suffix to be used for each name generated.
Storage class	A Automatic (default) S Static.
Validate	Determines how invalid characters are handled. blank Invalid characters in a field name will result in an error message but will not be modified. rep.char Invalid characters in a field name are replaced by this character. Valid values: A-Z, a-z, 0-9 and _ (underscore). * Invalid characters in a field name are deleted.
Upper or lower case	Case of names in generated code. U upper-case names L lower-case names.
Truncation	Specifies which characters are deleted if a generated field name is too long: L Truncate from the left R Truncate from the right M Truncate from the middle A warning is given if field names are truncated.
Record buffer name	Specifies the name of the record buffer in the generated structure. If omitted, the file ID is used.
Format buffer name	Specifies the name of the format buffer in the generated structure. If omitted, the file ID prefixed by F_ is used.

Generate C Include Code in Batch Mode

Command: GENERATE LANG-C

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Save as member	MEM	2
Save in library	LIB	3
Overwrite option	REPLACE	4
Op. system member	OS-MEMBER	5
Punch / output	PUNCH	6
List generated code	LIST	7
List offsets	OFFSET	8
Generate format buffer	FORMAT-BUFFER	9
Adabas version	ADA-VER	10
Check field name	CHECK-NAME	11
Shift increment	SHIFT-INC	12
Field name prefix	PREFIX	13
Nr. of abstract lines	NR	14
Field name suffix	SUFFIX	15
Storage class	STO-CLASS	16
Validate	VALIDATION	17
Upper or lower case	UPPER-LOWER	18
Truncation	TRUNCATION	19
Record buffer name	RECORD-BUFFER-NAME	20
Format buffer name	FORMAT-BUFFER-NAME	21
Workfile name (see note below)	WORKFILE-NAME	22
If Entire System Server is used		
- Database ID	NP-DBID	23
- Dataset	NP-DSNAME	24
- Volume	NP-VOLSER	25
- Library	NP-LIB	26
- Sublibrary	NP-SUBLIB	27
- Member type	NP-MEMTYPE	28
- VSAM catalog name	NP-VSAMCAT	29

Note:

You can not specify a value other than blank (' ') for WORKFILE-NAME on mainframes.

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

Names in C Include Code

The following rules apply to the generation of names for C include code.

- Field names are derived from the name of the corresponding Predict field object if no C field name synonym has been specified.
- The case of alpha characters in field and file names is determined by the parameter Upper or lower case.
- The parameters Field name prefix, Field name suffix, Validate and Truncate apply to the generation of field names (see description of parameters above).
- If supplementary fields have to be generated the names of these extra fields are derived from the Predict field names by generating prefixes. The rules for generating names for extra fields are described in the sections below.
- If a generated field name or type name is not unique, an error message may be displayed if Check field name is either set to A or Y.

C Names for Redefinition Fields - RE

When a field is redefined, a data type with structure union is generated. The name of the data type is tu_fieldname. The name of the corresponding variable is u_fieldname. The union consists of

- the redefined field
- one type tn for each redefinition with corresponding variable rn_fieldname.

C Names for Counter Fields (MC or PC)

When generating C fields for Predict fields of type MC or PC, a counter field will automatically be generated in the copy code. The name of this field is derived from the ID of the MC or PC field by adding the prefix C-.

With the following parameter settings

- Field name prefix = Adabas
- Field name suffix = *PERSONNELOFFICE
- Validate = _
- Truncation = R
- Max. name length = 30

the following field names are generated in C include code for field HOURS_DAY of type MC:

ADABAS_C_HOURS_DAY_PERSONNELOF ADABAS_HOURS_DAY_PERSONNELOFFI
--

C Names for Additionally Generated Indicator Fields

An additional indicator field prefixed with s_ is generated in the following cases:

- a field is defined with Suppression option set to U (null allowed)
- a field of a file of type A or U is defined with Suppression option set to R (not null)

Note:

For Adabas fields, the additional indicator field is only generated for the following Adabas versions:

- I3 or above
- U1 or above
- V4 or above.

Field Format and C Include Code

Fields in the C include code have a C clause determined by the length and format of the corresponding Predict field object, as shown in the table below:

Predict Format	File Type	Predict Length	C Clause	Note
B/I	YT, YV	1	CS_TINYINT	
I	JT, JV	1	short	
	BT, BV, XT, XV	2	short int	Irrespective of default value for Represent. of I2
	JT, JV, B, AT A(SQL)	2	short	A(SQL) means file type A with Adabas SQL usage=Y
	YT, YV	2	CS_SMALLINT	
B/I	other	2	short, int or unsigned char [2]	Depending on default value Represent. of I2.
I	BT, BV, XT, XV	4	long int	Irrespective of default value for Represent. of I4
	JT, JV	4	long	If in defaults Represent. of I4=L
			int	If in defaults Represent. of I4≠L
	B, AT A(SQL)	4	long	Irrespective of default value for Represent. of I4
	YT, YV	4	CS_INT	
B/I	other	4	int, long or unsigned char [4]	Depending on default value Represent. of I4.
		8	unsigned char [8]	
B	A(SQL), AT,B		ESQ-BINARY[l]	
	other	$l=3,5,6,7$	unsigned char [l]	
		$l \geq 9$	unsigned char [l]	
F	YT, YV	4	CS_REAL	
	other	4	float	
	YT, YV	8	CS_FLOAT	
	other	8	double	
NS/US	XT, XV	nn.m	dec_t	
	B, AT A(SQL)	nn.m	long	nn < 10, m=0
			double	all other values of n and m
	D, E	nn.m	DECIMAL (l,m)	$l=nn+m$
	BT, BV	nn.m	DECIMAL {l,m}	if $nn+m < 13$: $l=nn+m$
			char [l]	if $nn+m > 13$: $l=nn+m$
NS	JT, JV	nn.m	double	

N/U NS/US	other	nn.m	unsigned char [<i>l</i>]	<i>l</i> =nn+m
	D, E		DECIMAL (<i>l,m</i>)	
PS	JT, JV	nn.m	double	
	XT, XV	nn.m	dec_t	
	B, AT A(SQL)	nn.m	long	nn < 10, m=0
			double	all other values of nn and m
BT, BV	nn.m	DECIMAL { <i>l,m</i> }	if nn+m<13: <i>l</i> =nn+m	
		char [<i>l</i>]	if nn+m>13: <i>l</i> =(nn+m+2)/2	
P/PS	other	nn.m	unsigned char [<i>l</i>]	<i>l</i> =(nn+m+2)/2
	D, E		DECIMAL (<i>l,m</i>)	<i>l</i> =nn+m
D	D, E		unsigned char [10]	* See note at the end of this table
	BT, BV		char [8]	
	XT, XV		long int	
	other		unsigned char [4]	
T	BT, BV		char [8]	* See note at the end of this table
	D,E		unsigned char [8]	* See note at the end of this table
	other		unsigned char [7]	
L	BT, BV		short int	
	other		unsigned char [1]	
BT	YT, YV		CS_BIT	
DT	JT, JV		unsigned char [25]	* See note at the end of this table
	OT, OV		unsigned char [9]	
	XT, XV		dtime_t	
	YT, YV		CS_DATETIME	
DS	YT, YV		CS_DATETIME4	
G	D, E	n	unsigned char [<i>m</i>]	m=2*n
GV	D, E	n	unsigned char [<i>m</i>]	
GL	D, E	n	unsigned char [<i>m</i>]	
IV	XT, XV	7 17	intrvl_t	
S	XT, XV		long int	
TS	BT, BV		char [20]	* See note at the end of this table
	D, E		unsigned char [26]	* See note at the end of this table
	YT, YV		CS_VARBINARY	
MO	JT, JV		double	
	XT, XV	n.m	dec_t	
	YT, YV		CS_MONEY	

MS	YT, YV		CS_MONEY4	
TK	JT, JV			Is generated like format A with Character Set B and n=8. See next table below.
OK	JT, JV			As above, but n=16.

Note:

For fields marked with an asterisk (*), the length given in the table is applicable if the field is within a redefinition. If the field is not within a redefinition, an additional byte is used to denote the end of the string.

File Type in Predict	Predict Format	Predict Character Set	Length	C Clause	Note
BT, BV	A	any	n	char [n]	* See note above this table
	AL	any	n	VARCHAR [n]	
				VARCHAR *	
AV	any	n	VARCHAR [n]		
JT, JV	A, AV	blank	n	unsigned char [n]	* See note above this table
		B	n	Two-level group is generated: varchar struct { short Fieldname_len; char Fieldname_txt [n]; } Fieldname;	
	AL	B	n	Two-level group is generated: varchar struct { short Fieldname_len; char Fieldname_txt [n]; } Fieldname;	Field is skipped if no length is specified
B, BL, BV			n	Two-level group is generated: varbyte struct { short Fieldname_len; char Fieldname_txt [n]; } Fieldname;	Field is skipped if no length is specified

OT, OV	A, AL	blank	n	unsigned char [n]	
	AV	blank	n	VARCHAR [n]	
	A	B	n	RAW	
	AL	B	n	LONGRAW	
	LO	blank, M	n	OCICloblocator *	Indicator field is generated as OCIInd
		B	n	OCIBloblocator *	Indicator field is generated as OCIInd
XT, XV	A	blank	n	unsigned char [n]	* See note above this table
		M	n	Two-level group is generated: struct t_Fieldname { short int Fieldname_len; unsigned char Fieldname_txt [n]; } Fieldname;	
	AL	any	n	loc_t	
	AV	any	n	Two-level group is generated: struct t_Fieldname { short int Fieldname_len; unsigned char Fieldname_txt [n]; } Fieldname;	
YT, YV	A	S	n	CS_CHAR [n]	
	A	D	n	CS_CHAR [2*n]	
	A	B	n	CS_BINARY [2*n]	
	AL	blank		CS_TEXT	
	AL	B		CS_IMAGE	
	AV	S	n	CS_VARCHAR	
	AV	D	n	CS_VARCHAR	
	AV	B	n	CS_VARBINARY	

D, E	A, AL	any	n	unsigned char[n]	* See note above this table
	AV	any	n	Two-level group is generated: struct t_Fieldname { i2form Fieldname_len; unsigned char Fieldname_txt [n]; } Fieldname;	i2form corresponds to the code generated for an I2 field: short, int or unsigned char [2]
	LO	any	n	int, long or unsigned char[4]	Depending on default value Represent. of I4. (Field is generated as locator)
A(SQL), B, AT	A, AV	only 1 Char. Set	n	unsigned char [n]	* See note above this table
A, U	A	anyt	n	unsigned char [n]	
	AV		n	Two-level group is generated. See file type D.	

An automatically generated counter field has the same clause as an I2 field.

A numeric or binary format field with a length not included in the table above is treated in C include code as an alphanumeric format field. A warning message is given.

Note:

If C include code for DB2 tables/views is generated, any redefinition of a field with format NS or US is skipped.

Sample Output

```
13:27:28          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Generate C Copy Code -                    Page: 1
```

```
File ID .. GENERATION-EXAMPLE
```

```

/*****
/*      THIS RECORD-BUFFER LAYOUT WAS GENERATED BY PREDICT      */
/*      FOR FILE: GENERATION-EXAMPLE                            */
/*      ON: 2003-05-31 STARTING AT 13:27:28                    */
/*      FILE-COMMENTS: Example file for the                    */
/*      PREDICT generation subsystem                            */
/*      ..                                                       */
/*****
struct t_generation_example
{
    struct t_group_1
    {
        unsigned char ele_n_9v5[14];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' N ' --> ' A ' ) */
        struct t_gr_in_group
        {
            long ele_b_4;
            unsigned char ele_ps_5v2[4];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ' ) */
            union tu_mu_b_4
            {
                long mu_b_4[5];
                struct t1_mu_b_4
                {
                    unsigned char mu_red_b_7[7];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' ) */
                    unsigned char mu_red_us_13[13];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' NS ' --> ' A ' ) */
                } r1_mu_b_4;
                struct t2_mu_b_4
                {
                    unsigned char mu_red_b_8[8];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' ) */
                } r2_mu_b_4;
            } u_mu_b_4;
        } gr_in_group;
        unsigned char ele_a_42[42];
        double ele_f_8;
        unsigned char ele_b_3[3];
    }
}

```

```

13:27:28          ***** P R E D I C T 4.3.1 *****          2003-05-31
                  - Generate C Copy Code -                      Page: 2

File ID .. GENERATION-EXAMPLE

/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' ) */
    } group_1;
    short c_pc_occ_7;
    struct t_pc_occ_7
    {
        unsigned char pc_ele_de_ns_7v3[10];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' NS ' --> ' A ' ) */
        short c_pc_mc_ps_6v1;
        unsigned char pc_mc_ps_6v1[11] [4];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ' ) */
        struct t_pc_gr
        {
            short pc_ele_i_2;
            unsigned char pc_ps_20v7[14];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ' ) */
        } pc_gr;
        float pc_ele_f_4;
        } pc_occ_7[7];
        unsigned char ele_d[4];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' D ' --> ' A ' ) */
        unsigned char ele_t[7];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' T ' --> ' A ' ) */
        unsigned char ele_l;
        } generation_example;
/*****
/*      THIS FORMAT-BUFFER WAS GENERATED BY PREDICT
/*      FOR FILE: GENERATION-EXAMPLE
/*      ON: 2003-05-31 STARTING AT 13:27:28
/*****
unsigned char f_generation_example [ 293 ] =
    "AB,2X,AD,AE,AF1-5,AG,2X,AH,AI,1X,ALC,2,AM1,AN1C,2,AN"
    "1(1-11),AO1,2X,AR1,2X,AM2,AN2C,2,AN2(1-11),AO2,2X,AR"
    "2,2X,AM3,AN3C,2,AN3(1-11),AO3,2X,AR3,2X,AM4,AN4C,2,A"
    "N4(1-11),AO4,2X,AR4,2X,AM5,AN5C,2,AN5(1-11),AO5,2X,A"
    "R5,2X,AM6,AN6C,2,AN6(1-11),AO6,2X,AR6,2X,AM7,AN7C,2,"
    "AN7(1-11),AO7,2X,AR7,2X,AS,AT,AU."
/*****

DIC1800 SUMMARY:      25 FIELD(S) PROCESSED
DIC1818 WARNING:      11 FORMAT(S) CHANGED
DIC1819 MESSAGE:      33 SLACK BYTE(S) GENERATED
DIC1847 MESSAGE:       3 FIELD(S) SKIPPED FOR RECORDBUFFER STRUCTURE

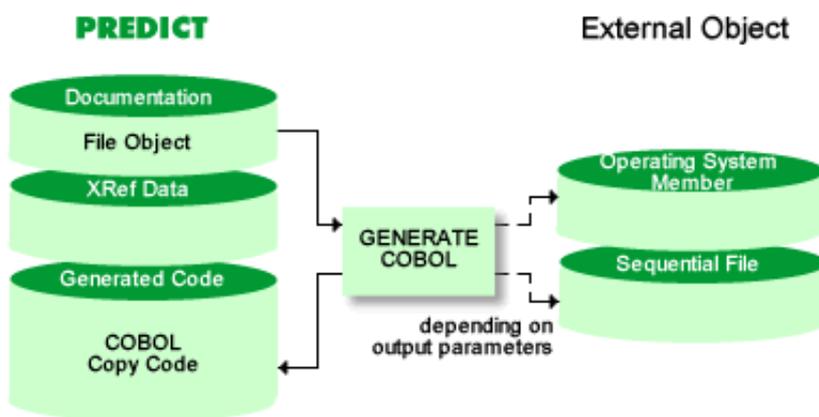
```

COBOL Copy Code

The function Generate COBOL Copy Code generates a record buffer for use in COBOL programs based on a Predict file object.

In addition, an Adabas format buffer can also be generated if required for files of the following types:

- Adabas file (file type A) with parameter Adabas SQL usage = N
- Adabas userview (file type U)



This section contains:

- Calling the Function
- Generate COBOL Copy Code in Batch Mode
- Names in COBOL Copy Code
- Field Format and COBOL Copy Code
- COBOL Copy Code for DB2 Tables/Views
- Sample Output

Calling the Function

The Generate COBOL Copy Code screen is displayed with function code G and object code CO in a Predict main menu, or with the command GENERATE COBOL.

```

10:03:05          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Generate COBOL Copy Code -

File ID .....*
Save as member .....          Save in library .... COBLIB
Overwrite option ..... Y (Y,N) Op. system member ..
Punch / output .....* N          List offsets .....* N
List generated code ..... Y (Y,N) Adabas version ....* I7
Generate format buffer ....* N          Field name prefix ..
Check field name .....* A          Field name suffix ..
Start level ..... 1 (0-40)          Validate ..... -
Level number increment ..... 1 (1-40) Truncation .....* R
Level shift increment ..... 3 (0-9) With Cond. names ... N (Y,N)
Nr. of abstract lines ..... 3 (0-16) Indexed by .....* N
Generate initial value ....* N          Literal delimiter .* S
Synchronized .....* Y          Decimal character .* P
Depending on ..... N (Y,N)          Redefinition name .* S
Record buffer name .....
Format buffer name .....

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presettings	
The parameters below can be changed in the Modify COBOL Defaults screen. See Generation Defaults.	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated. See Parameters Specifying the Form of Output for more information.
Compiler	Copy code can be generated for the following compilers: 7 Standard COBOL 74 compiler. 8 COBOL 85 compiler. W COBOL compiler for WANG.
Parameters	
The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output, Workfile name and List generated code are described in the section Parameters Specifying the Form of Output.	
File ID	ID of the Predict file object from which the definitions are to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.

List offsets	<p>Y The offset of each item in the record buffer structure (relative to the beginning of the structure) in decimal and hexadecimal formats is to be included as a comment. The total length of each buffer is also included.</p> <p>P As above, but the absolute position (offset+1) is included as a comment.</p> <p>L The total lengths of the record buffer and the format buffer are to be included as a comment.</p> <p>V The file number and the calculated lengths of the record buffer and the format buffer are to be generated as constants in the copy code. The name of the file number constant is the record buffer name prefixed by N-. The name of each length constant is the appropriate buffer name prefixed by L-. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.</p> <p>N No offset.</p>
Adabas version	The version of Adabas for which the copy code is to be generated. Enter an asterisk for valid values or see table in the section Adabas Version for more information.
Generate format buffer	<p>The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas SQL usage set to N) or for files of type U.</p> <p>Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, set this parameter to F or N.</p>
Field name prefix	The prefix to be used for each field name generated.
Check field name	<p>A COBOL field names are checked for uniqueness within the whole structure.</p> <p>Y Structure levels are included in the validation check of the field names: if two fields have the same name, they must be separated by at least one field with a different name and a smaller level number.</p> <p>N No check for duplicate field names is performed.</p>
Field name suffix	The suffix to be used for each field name generated.

Start level	The starting level number to be used for the generated statements.												
Validate	<p>Determines how invalid characters are handled.</p> <p>blank Invalid characters in a field name will result in an error message but will not be modified.</p> <p>rep.char Invalid characters in a field name are replaced by this character. Valid replace characters: letters A-Z, digits 0-9 and hyphen (-).</p> <p>*</p> <p>Invalid characters in a field name are deleted.</p>												
Level number increment	<p>This parameter is evaluated with the parameter Start level. The level numbers of the generated statements are incremented as follows</p> <table border="1"> <thead> <tr> <th>Start Level</th> <th>Level number increment (example)</th> <th>Level numbers</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>5</td> <td>1, 6, 11...</td> </tr> <tr> <td>1</td> <td>5</td> <td>1, 5, 10 ...</td> </tr> <tr> <td>2 - 40</td> <td>5</td> <td>n, n+5, n+10...</td> </tr> </tbody> </table>	Start Level	Level number increment (example)	Level numbers	0	5	1, 6, 11...	1	5	1, 5, 10 ...	2 - 40	5	n, n+5, n+10...
Start Level	Level number increment (example)	Level numbers											
0	5	1, 6, 11...											
1	5	1, 5, 10 ...											
2 - 40	5	n, n+5, n+10...											
Truncation	<p>Specifies which characters are deleted if a generated field name is too long:</p> <p>L truncate from the left</p> <p>R truncate from the right</p> <p>M truncate from the middle</p> <p>A warning is given if field names are truncated.</p>												
Level shift increment	The number of positions to be shifted right when a level higher than the current level number is found.												
With Cond. names	<p>Y Any condition names defined in the Predict field objects are to be generated on level 88, provided that the respective field objects have one of the following formats:</p> <ul style="list-style-type: none"> ● A All lengths ● N or P Less than 19 digits ● I or B 2, 4 or 8 digits ● L A FALSE-condition will always be generated. The Condition name is then generated by concatenating the field name to the prefix N- (if not specified explicitly). <p>These names are prefixed, suffixed, validated and truncated in the same way as field names.</p>												
Nr. of abstract lines	The number of Predict abstract lines per field to be included in the generated code.												

<p>Indexed by</p>	<p>Y COBOL clause INDEXED BY is generated for all repetitive fields (MC, MU, PC and PE). For a field which has no INDEXED BY name in its Predict object, I- followed by the field name is used as index name. These names are prefixed, suffixed, validated and truncated in the same way as field names.</p> <p>N COBOL clause INDEXED BY will not be generated for any field.</p> <p>S COBOL clause INDEXED BY is generated only for repetitive fields which have an INDEXED BY name in their Predict objects.</p>																		
<p>Generate initial value</p>	<p>Y Statements are generated to initialize the structure with the value for Init value defined for the field objects in Predict. Fields with no value defined for Init value are initialized as follows</p> <table border="1" data-bbox="391 801 901 1099"> <thead> <tr> <th>Format</th> <th>Length</th> <th>Initialized with</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>all lengths</td> <td>spaces</td> </tr> <tr> <td>N, NS, P, PS</td> <td>< 19 digits</td> <td>zeros</td> </tr> <tr> <td>I, B</td> <td>2,4,or 8 digits</td> <td>zeros</td> </tr> <tr> <td>D, T</td> <td>-</td> <td>zeros</td> </tr> <tr> <td>Logical</td> <td>-</td> <td>low value</td> </tr> </tbody> </table> <p>No members of redefine groups are initialized.</p> <p>N No initialization.</p> <p>S Only fields which have a value defined for Init value in the corresponding Predict object are initialized.</p>	Format	Length	Initialized with	A	all lengths	spaces	N, NS, P, PS	< 19 digits	zeros	I, B	2,4,or 8 digits	zeros	D, T	-	zeros	Logical	-	low value
Format	Length	Initialized with																	
A	all lengths	spaces																	
N, NS, P, PS	< 19 digits	zeros																	
I, B	2,4,or 8 digits	zeros																	
D, T	-	zeros																	
Logical	-	low value																	
<p>Literal delimiter</p>	<p>Literal delimiter character:</p> <p>D double quotation marks (")</p> <p>S single quotation marks (').</p>																		
<p>Synchronized</p>	<p>Determines which fields are to be aligned. Fields are aligned in the record buffer and in the format buffer on the boundary shown in the table in the section Field Format and COBOL Copy Code.</p> <p>Y All fields with COBOL attribute COMP, COMP-1 or COMP-2 are aligned.</p> <p>N No fields are aligned.</p> <p>S Fields are aligned only if the 3GL specification Synchronized of the corresponding Predict object is set to S.</p>																		

Decimal character	<p>P decimal point</p> <p>C decimal comma</p>
Depending on	<p>Y COBOL attribute OCCURS DEPENDING ON is generated for a field or field group in a file if it has type PE or MU and a DEPENDING ON name is specified for this field.</p> <p>These names are prefixed, suffixed, validated and truncated in the same way as field names.</p> <p>Note: This option is not allowed for files of type A or U. This option is ignored when using a WANG COBOL compiler.</p>
Redefinition name	<p>Determines how COBOL field names for Predict fields of type RE are generated:</p> <p>F The string FILLER is used as redefinition name.</p> <p>S The suffix REGR is added to the Predict field name. If a field name is redefined more than once, the suffix will have the form REGRn, where n is an integer incremented by 1 for each field of type RE.</p>
Record buffer name	Specifies the name of the record buffer in the generated structure. If omitted, the file ID is used.
Format buffer name	Specifies the name of the format buffer in the generated structure. If omitted, the file ID prefixed by FORMAT-BUFFER- is used.

Generate COBOL Copy Code in Batch Mode

Command: GENERATE COBOL

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Save as member	MEM	2
Save in library	LIB	3
Overwrite option	REPLACE	4
Op. system member	OS-MEMBER	5
Start level	START-LEVEL	6
Field name prefix	PREFIX	7
Level number increment	LEVEL-INCREMENT	8
Field name suffix	SUFFIX	9
Level shift increment	LEVEL-SHIFT	10
Nr. of abstract lines	NR	11

Validate	VALIDATION	12
Generate initial value	INIT	13
Truncation	TRUNCATION	14
Generate format buffer	FORMAT-BUFFER	15
List offsets	OFFSET	16
List generated code	LIST	17
Punch / output	PUNCH	18
Record buffer name	RECORD-BUFFER-NAME	19
Format buffer name	FORMAT-BUFFER-NAME	20
With Cond. names	COND-NAME	21
Synchronized	SYNC	22
Indexed by	INDEXED	23
Depending on	DEPENDING	24
Adabas version	ADA-VER	25
Check field name	CHECK-NAME	26
Literal delimiter	DELIMITER	27
Decimal character	DECIMAL-CHAR	28
Redefinition name	REDEFINE-NAME	29
Workfile name (see note below)	WORKFILE-NAME	30
If Entire System Server is used		
- DB-ID	NP-DBID	31
- Dataset	NP-DSNAME	32
- Volume	NP-VOLSER	33
- Library	NP-LIB	34
- Sublibrary	NP-SUBLIB	35
- Member type	NP-MEMTYPE	36
- VSAM catalog	NP-VSAMCAT	37

Note:

You can not specify a value other than blank (' ') for WORKFILE-NAME on mainframes.

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

Names in COBOL Copy Code

The following rules apply to the generation of names for COBOL copy code.

- Field names are derived from the name of the corresponding Predict field object if no COBOL field name synonym has been specified.
- Alphabetic characters of field and file names are converted to upper-case. The parameters Field name

prefix, Field name suffix, Validate and Truncate apply to the generation of field names (see description of parameters above).

- In cases when supplementary fields have to be generated the names of these extra fields are derived from the Predict field names by generating prefixes and/or suffixes.
- If a resulting field name, indexed by name or depending on name is not unique, an error message may be displayed if Check field name is either set to A or Y.

The rules applying when deriving the names of extra fields when generating COBOL copy code are described in the sections below.

COBOL Names for Counter Fields - MC or PC

When generating COBOL fields for Predict fields of type MC or PC, a counter field will automatically be generated in the copy code. The name of this field is derived from the name of the MC or PC field by adding the prefix C-.

COBOL Names for Multiple Fields - MU - Contained in Periodic Groups - PE

When generating COBOL fields for a Predict field of type PE that contains a field of type MU and the parameter Gr.structur is set to N, a dummy field will automatically be generated preceding the MU field. The name of this field is derived from the name of the MU field by adding the prefix G-.

The MU field can then be used as a two-dimensional table and its real occurrence need not be computed.

COBOL Names for Redefined Fields - RE in Periodic Groups - PE

When generating COBOL fields for a Predict field of type PE that contains a redefinition and the parameter Gr.structur is set to N, a dummy field will automatically be generated preceding the RE field. The name of this field is derived from the name of the RE field by adding the prefix R-.

COBOL Names for Logical Fields - L

When generating COBOL fields for a Predict logical field (field format L) with the parameter With cond. names set to Y and no condition name is defined for the field, a condition name is generated by adding the prefix N- to the field ID.

COBOL Names for Record Buffer and Format Buffer

The record buffer name and the format buffer name are generated without prefix and suffix. The value(s) supplied must not contain invalid characters. If the file ID is taken as default record buffer name or, with the prefix FORMAT-BUFFER-, as default format buffer name, it is validated and, if necessary, truncated.

COBOL Names for Default Indexed by Name

When COBOL copy code is generated with the option Indexed by set to Y, and no INDEXED BY NAME has been specified, this name is generated by adding the prefix I- to the Predict name.

COBOL Names for Additionally Generated Indicator Fields

An additional indicator field prefixed with S- is generated in the following cases:

- a field is defined with Suppression option set to U (null allowed)
- a field of a file of type A, U or B is defined with Suppression option set to R (not null)

Note:

For Adabas fields, the additional indicator field is only generated for the following Adabas versions:

- I3 or above
- U1 or above
- V4 or above.

Examples for the Generation of COBOL Field Names

The GENERATE COBOL function is applied on a field HOURS_DAY. This field has the following attributes:

- A field name synonym for COBOL is not defined.
- The field type is MC.
- The field occurs in a periodic group.
- The parameter Gr.structur is set to N.

Three fields are generated in the COBOL copy code

```

C-HOURS_DAY
G-HOURS_DAY
HOURS_DAY
```

If the parameter Field name prefix had been set to Adabas, the parameter Field name suffix to *PERSONNELOFFICE, Validate to -, Truncation to R and Indexed by to Y when generating COBOL copy code from this field, the following COBOL field names are generated:

```

ADABAS-C-HOURS-DAY-PERSONNELOF indexed by ADABAS-I-C-HOURS-DAY-PERSONNEL
ADABAS-G-HOURS-DAY-PERSONNELOF indexed by ADABAS-I-G-HOURS-DAY-PERSONNEL
ADABAS-HOURS-DAY-PERSONNELOFFI indexed by ADABAS-I-HOURS-DAY-PERSONNELOF
```

Field Format and COBOL Copy Code

Fields in the COBOL copy code have a COBOL clause determined by the following factors:

- length and format of the corresponding Predict field object
- file type of the file containing the field
- the compiler for which you are generating.

File type A(SQL) in the table below represents files of type A with parameter Adabas SQL usage set of Y.

Predict Format	Predict Length	File Type or Compiler	COBOL Clause	Note
B	l=1,3,5,6,7		PIC X(l)	
I	1	File Type: JT, JV	PIC S9(2) COMP	
		other file types	PIC X(1)	

B/I	2	File Type: BT, BV or Compiler: WANG/VS (and file type not JT, JV)	PIC S9(4) BINARY	aligned on half word boundary (if word length=4)
		File Type: JT, JV or Compiler: 74,85 (and file type not BT, BV)	PIC S9(4) COMP	
	4	File Type: BT, BV or Compiler: WANG/VS (and file type not JT, JV)	PIC S9(9) BINARY	aligned on word boundary (if word length=4)
		File Type: JT, JV or Compiler: 74,85 (and file type not BT, BV)	PIC S9(9) COMP	
	8	Compiler: WANG/VS	PIC X(8)	
		other compiler	PIC S9(18) COMP	aligned on word boundary (if word length=4)
B	$l > 9$		PIC X(l)	
F	$l = 4, 8$	File Type: not BT, BV, JT, JV Compiler: WANG/VS	PIC X(l)	
		File Type: JT, JV	PIC S9(10)V9(8) COMP-3	
	4	File Type: BT, BV or Compiler: not WANG/VS	COMP-1	aligned on word boundary (if word length=4)
	8	File Type: BT, BV or Compiler: not WANG/VS	COMP-2	aligned on double word boundary (if word length=4)
N/U	$nn.m$ ($nn+m < 19$)		PIC 9(nn)V9(m)	if $m=0$, V9(m) is not applicable
	$nn.m$ ($nn+m > 18$)		PIC X($nn+m$)	
NS/US	$nn.m$ ($nn+m > 15$)	File Type: B, A(SQL), AT	PIC X($nn+m$)	
	$nn.m$ ($nn+m < 19$)		PIC S9(nn)V9(m)	if $m=0$, V9(m) is not applicable
	$nn.m$ ($nn+m > 18$)		PIC X($nn+m$)	

P	nn.m (nn+m<19)	Compiler: 74	PIC 9(nn)V9(m)COMP-3	if m=0, V9(m) is not applicable
		Compiler: 85	PIC 9(nn)V9(m)PACKED DECIMAL	
		Compiler: WANG	PIC 9(nn)V9(m)COMP	
	nn.m (nn+m>18)		PIC X(l)	$l=(nn+m+2)/2$
PS	nn.m (nn+m>15)	File Type: B, A(SQL) AT	PIC X(l)	$l=(nn+m+2)/2$
		File Type: not BT, BV or Compiler: 85	PIC S9(nn)V9(m) PACKED-DECIMAL	if m=0, V9(m) is not applicable
	File Type: not BT, BV or Compiler: 74	PIC S9(nn)V9(m)COMP-3		
	File Type: not BT, BV or Compiler: WANG	PIC S9(nn)V9(m)COMP		
	nn.m (nn+m>18)		PIC X(l)	$l=(nn+m+2)/2$
D		File Type: D, E, BT, BV	PIC X(10)	
		File Type: XT, XV	see format I, length 4	
		other file types	see Format P, length 6	
T		File Type: D, E, BT, BV	PIC X(8)	
		other file types	see Format P, length 12	
L		File Type: BT, BV	PIC S9(4) BINARY	
		other file types	PIC X(1)	
BT		File Type: YT, YV	PIC X(1)	
DT		File Type: JT, JV	PIC X(25)	
		File Type: OT, OV	PIC X(9)	
		File Type: XT, XV	PIC X(23)	
		File Type: YT, YV	PIC X(32)	
DS		File Type: YT, YV	PIC X(28)	
G, GV, GL	n	File Type: D, E	PIC X(m)	$m=2*n$
IV	7 / 17	File Type: XT, XV	PIC X (n)	
S		File Type: XT, XV	see format I, length 4	
TK		File Type: JT, JV	PIC X(8)	

TS		File Type: BT, BV, D, E	PIC X(26)	
		File Type: YT, YV	PIC X(8)	
OK		File Type: JT, JV	PIC X(16)	
MO		File Type: JT, JV, YT, YV	see format F, length 8	
	n.m	File Type: XT, XV	PIC S9(n)V9(m)	
MS		File Type: YT, YV	see format F, length 4	

A numeric or binary format field with a length not included in the table above is treated in COBOL copy code as an alphanumeric format field and cannot have an initial value generated for it. A warning message is issued.

File Type in Predict	Predict Format	Predict Char Set	Length	COBOL Clause	Note
YT, YV	A	S	n	PIC X(n)	
		B	n	PIC X(m)	m=2*n
		D	n	PIC X(m)	m=2*n
	AV	S	n	Two-level group is generated: 02 Fieldname	m=n
		B, D		03 Fieldname-LEN corresponding to I2 03 Fieldname-TXT Char of length m	m=2*n
JT, JV	A, AV	any	n	PIC X(n)	
	AL	B	n	PIC X(n)	Field is skipped if no length is specified
	BL		n	PIC X(n)	Field is skipped if no length is specified
	BV		n	PIC X(n)	
XT, XV, OT, OV,	A, AL	any	n	PIC X(n)	
	AV	there is only one Char Set	n	Two-level group is generated: 02 Fieldname 03 Fieldname-LEN corresponding to I2 03 Fieldname-TXT Char of length n	

D, E	A, AL	any	n	PIC X(n)	
	AV	any	n	Two-level group is generated: 02 Fieldname 03 Fieldname-LEN corresponding to I2 03 Fieldname-TXT Char of length n	
A, U	A	there is only one Char Set	n	PIC X(n)	
	AV		n	Two-level group is generated: 02 Fieldname 03 Fieldname-LEN corresponding to I2 03 Fieldname-TXT Char of length n	
A(SQL), AT, B	A, AV	there is only one Char Set	n	PIC X(n)	
BT, BV	A	any	n	PIC X(n)	
	AL	any	n	Two-level group is generated: 02 Fieldname 03 Fieldname-LEN corresponding to I2 03 Fieldname-TXT Char of length n	Field is skipped if no length is specified
	AV	any	n	Two-level group is generated: 02 Fieldname 03 Fieldname-LEN corresponding to I2 03 Fieldname-TXT Char of length n	

Automatically Generated Counter Fields

An automatically generated counter field has the following clause, depending on the COBOL compiler used:

Compiler	Clause
COBOL 74, COBOL 85	PIC S9(4) COMP
WANG	PIC S9(4) BINARY

COBOL Copy Code for DB2 Tables/Views

If COBOL copy code for DB2 Tables/Views is generated, the format NS or US is converted to the following COBOL format depending on the COBOL compiler used:

Compiler	Clause
COBOL 74,	PIC S9(nn)V9(m) COMP-3
COBOL 85	PIC S9(nn)V9(m) COMP
WANG	PIC S9(nn)V9(m) PACKED-DECIMAL

Any redefinitions that exist for this field are skipped.

A DB2 large object is generated as locator like an I4 field.

Sample Output

```

13:27:53          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate COBOL Copy Code -                   Page: 1

File ID .. GENERATION-EXAMPLE

*****
*      THIS RECORD-BUFFER LAYOUT WAS GENERATED BY PREDICT
*      FOR FILE: GENERATION-EXAMPLE
*      ON: 2003-05-31 STARTING AT 13:27:53
*      FILE-COMMENTS: Example file for the
*      PREDICT generation subsystem.
*
*****

01  GENERATION-EXAMPLE.
   02  GROUP-1.
      03  ELE-N-9V5          PIC 9(9)V9(5).
      03  GR-IN-GROUP.
         04  ELE-B-4          PIC S9(9) COMP SYNC.
            88  INIT-ELE-B-4  VALUE IS 4096.
            88  COND-ELE-B-4-V-1
               VALUE IS -1.
         04  ELE-PS-5V2      PIC S9(5)V9(2) COMP-3.
         04  R-MU-B-4.
            05  MU-B-4          OCCURS 5 TIMES PIC S9(9) COMP
                               SYNC.
         04  R-MU-B-4-REGR   REDEFINES R-MU-B-4.
            05  MU-RED-B-7    PIC X(7).
* >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' )
         05  MU-RED-US-13    PIC S9(13).
         04  R-MU-B-4-REGR1  REDEFINES R-MU-B-4.
            05  MU-RED-B-8    PIC S9(18) COMP.
               88  MU-RED-B-8-COND1
                  VALUE IS 1118481.
         03  ELE-A-42        PIC X(42).
            88  COND-NAME-START VALUE IS 'COND-START' THRU
                  'COND-ZEND'.
            88  FILL-WITH-STRING VALUE IS ALL 'A'.
         03  ELE-F-8         COMP-2 SYNC.
         03  ELE-B-3         PIC X(3).

```

```
13:27:53          ***** P R E D I C T 4.3.1 *****          2003-05-31
                  - Generate COBOL Copy Code -                    Page: 2
```

```
File ID .. GENERATION-EXAMPLE
```

```
* >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' )
```

```
02 C-PC-OCC-7          PIC S9(4) COMP SYNC.
02 PC-OCC-7           OCCURS 7 TIMES.
03 PC-ELE-DE-NS-7V3   PIC S9(7)V9(3).
03 C-PC-MC-PS-6V1    PIC S9(4) COMP SYNC.
03 PC-MC-PS-6V1      OCCURS 11 TIMES PIC S9(6)V9(1)
                     COMP-3.

03 PC-GR.
04 PC-ELE-I-2        PIC S9(4) COMP SYNC.
04 PC-PS-20V7       PIC X(14).
```

```
* >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ' )
```

```
03 PC-ELE-F-4        COMP-1 SYNC.
02 ELE-D            PIC 9(6) COMP-3.
02 ELE-T            PIC 9(12) COMP-3.
02 ELE-L            PIC X(1).
88 N-ELE-L          VALUE IS LOW-VALUE.
```

```
*****
```

```
* THIS FORMAT-BUFFER WAS GENERATED BY PREDICT
```

```
* FOR FILE: GENERATION-EXAMPLE
```

```
* ON: 2003-05-31 STARTING AT 13:27:53
```

```
*****
```

```
01 FORMAT-BUFFER-GENERATION-EXAMP.
```

```
* >>> DIC1809 WARNING: FIELDNAME(S) TRUNCATED
```

```
02 FILLER          PIC X(54) VALUE IS
'AB,2X,AD,AE,AF1-5,AG,2X,AH,AI,1X,ALC,2,AM1,AN1C,2,AN1(' .
02 FILLER          PIC X(54) VALUE IS
'1-11),AO1,2X,AR1,2X,AM2,AN2C,2,AN2(1-11),AO2,2X,AR2,2X' .
02 FILLER          PIC X(54) VALUE IS
',AM3,AN3C,2,AN3(1-11),AO3,2X,AR3,2X,AM4,AN4C,2,AN4(1-1' .
02 FILLER          PIC X(54) VALUE IS
'1),AO4,2X,AR4,2X,AM5,AN5C,2,AN5(1-11),AO5,2X,AR5,2X,AM' .
02 FILLER          PIC X(54) VALUE IS
'6,AN6C,2,AN6(1-11),AO6,2X,AR6,2X,AM7,AN7C,2,AN7(1-11),' .
02 FILLER          PIC X(23) VALUE IS
'AO7,2X,AR7,2X,AS,AT,AU.' .
```

```
* *****
```

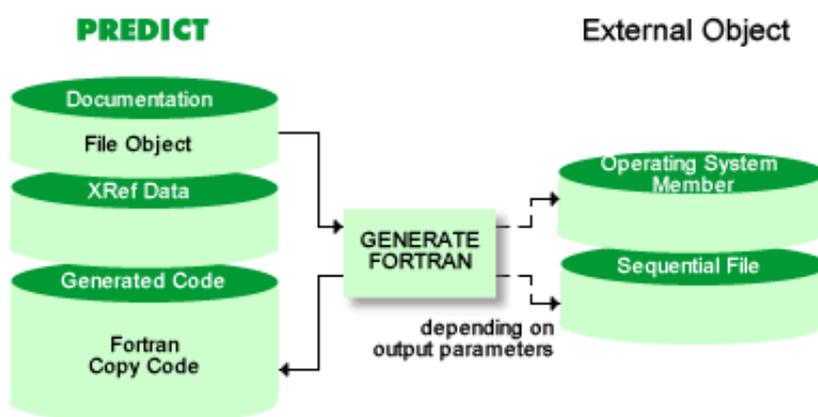
```
DIC1800 SUMMARY:    25 FIELD(S) PROCESSED
DIC1809 WARNING:    1 FIELDNAME(S) TRUNCATED
DIC1818 WARNING:    3 FORMAT(S) CHANGED
DIC1819 MESSAGE:    33 SLACK BYTE(S) GENERATED
DIC1847 MESSAGE:    3 FIELD(S) SKIPPED FOR RECORDBUFFER STRUCTURE
```

FORTRAN Copy Code

The function Generate FORTRAN Copy Code generates a record buffer for use in FORTRAN programs based on a Predict file object.

In addition, an Adabas format buffer can also be generated if required for files of the following types:

- Adabas file (file type A) with parameter Adabas SQL usage = N
- Adabas userview (file type U)



This section contains:

- Calling the Function
- Generate FORTRAN Copy Code in Batch Mode
- Names in FORTRAN Copy Code
- Field Format and FORTRAN Copy Code
- Sample Output

Calling the Function

The Generate FORTRAN Copy Code screen is displayed with function code G and object code FO in a Predict main menu, or with the command GENERATE FORTRAN.

```

10:13:24          ***** P R E D I C T  4.3.1  *****                2003-05-31
Plan    0          - Generate FORTRAN Copy Code -

File ID .....*

Save as member .....                               Save in library ..... FORLIB
Overwrite option ..... Y (Y,N)                    Op. system member .....
Punch directly .....* N

List generated code ..... Y (Y,N)                  Generate length.field .. N (Y,N)
Generate format buffer ....* N                     Adabas version .....* I7

Nr. of abstract lines ..... 4 (0-16)              Field name prefix .....
Generate initial value ....* N                     Field name suffix .....
Truncation .....* R                                Validate ..... *
Offset name ..... OFF
Record buffer name .....                           Compiler .....* I
Format buffer name .....

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presettings	
The parameters below can be changed in the Modify FORTRAN Defaults screen. See Generation Defaults.	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated. See Parameters Specifying the Form of Output for more information.
Parameters	
The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output, Workfile name and List generated code are described in the section Parameters Specifying the Form of Output.	
File ID	ID of the Predict file object from which the definitions are to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.
Generate length field	Y An INTEGER*2 field is to be created for each buffer generated. The initial value of this field is the length of the buffer (including slack bytes). Another INTEGER*2 field is created in the record buffer and will contain the file number. The name of the file number constant is the record buffer name prefixed by N. The name of each length constant will be the appropriate buffer name prefixed by L. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.

Generate format buffer	<p>The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas SQL usage set to N) or for files of type U.</p> <p>Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, set this parameter to F or N.</p>
Adabas version	The version of Adabas for which the copy code is to be generated. Enter an asterisk for valid values or see table in the section Adabas Version for more information.
Nr. of abstract lines	The number of Predict abstract lines per field to be included in the generated code.
Field name prefix	The prefix to be used for each field name generated.
Generate initial value	<p>Y Statements are generated to initialize the structure with the value for Init value defined for the field objects in Predict. Fields with no value defined for Init value are initialized with zeros or spaces.</p> <p>S Only fields which have a value defined for Init value in the corresponding Predict object are initialized.</p> <p>N No initialization.</p>
Field name suffix	Suffix appended to each field name generated.
Truncation	<p>Names are truncated to 30 characters for the VAX/VMS compiler. They are truncated to 6 characters for the IBM or Siemens compiler and FORTRAN 77 Standard.</p> <p>Specifies which characters are deleted if a generated field name is too long:</p> <p>L truncate from the left</p> <p>R truncate from the right</p> <p>M truncate from the middle</p> <p>A warning is given if field names are truncated.</p>

Validate	<p>Determines how invalid characters are handled.</p> <p>blank Invalid characters in a field name will result in an error message but will not be modified.</p> <p>rep.char Invalid characters in a field name are replaced by this character. Valid replace characters: letters A-Z, digits 0-9 and, for VAX/VMS only, \$ and _ (underscore).</p> <p>*</p> <p>Invalid characters in a field name are deleted.</p>
Offset name	<p>To display the offset of each item in the structure, Predict creates a CHARACTER*1 array with as many elements as there are bytes in the buffer, and generates EQUIVALENCE statements. Valid characters of this parameter are used as the name of the array in the record buffer. Valid characters of this parameter followed by AA are used as the name of the array in the format buffer.</p>
Record buffer name	<p>Specifies the name of the record buffer in the generated structure. If omitted, the file ID is used.</p>
Format buffer name	<p>Specifies the name of the format buffer in the generated structure. If omitted, the file ID prefixed by F is used.</p>
Compiler	<p>FORTRAN copy code can be generated for the following compilers:</p> <p>I IBM</p> <p>S Siemens</p> <p>V VMS</p> <p>7 FORTRAN 77 Standard</p>

Generate FORTRAN Copy Code in Batch Mode

Command: GENERATE FORTRAN.

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Save as member	MEM	2
Save in library	LIB	3
Overwrite option	REPLACE	4
Op. system member	OS-MEMBER	5
Nr. of abstract lines	NR	6
Field name prefix	PREFIX	7
Generate format buffer	FORMAT-BUFFER	8
Field name suffix	SUFFIX	9
Generate initial value	INIT	10
Generate length field	LENGTH-FIELD	11
List generated code	LIST	12
Validate	VALIDATION	13
Punch / output	PUNCH	14
Truncation	TRUNCATION	15
Offset name	OFFSET-NAME	16
Record buffer name	RECORD-BUFFER-NAME	17
Format buffer name	FORMAT-BUFFER-NAME	18
Adabas version	ADA-VER	19
Compiler	COMPILER	20
Workfile name (see note below)	WORKFILE-NAME	21
If Entire System Server is used		
- DB-ID	NP-DBID	22
- Dataset	NP-DSNAME	23
- Volume	NP-VOLSER	24
- Library	NP-LIB	25
- Sublibrary	NP-SUBLIB	26
- Member type	NP-MEMTYPE	27
- VSAM catalog	NP-VSAMCAT	28

Note:

You can not specify a value other than blank (' ') for WORKFILE-NAME on mainframes.

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

Names in FORTRAN Copy Code

The following rules apply to the generation of names for FORTRAN copy code.

- Alphabetic characters (letters) in Predict object IDs, such as file and field IDs, are converted into upper-case.
- Field names in FORTRAN copy code are created from Predict fields: either the field name synonym for FORTRAN or, if none exists, the Predict field ID.
- Any periodic groups with Gr.structur set to blank are treated as periodic groups with Gr.structur set to N and a warning message is issued.
- The parameters Validate and Truncate are applied when generating all field names:
 - when using compiler option IBM, Siemens, or FORTRAN 77, the names are truncated to 6 characters
 - with compiler option VMS, the names are truncated to 30 characters.
- Parameters Field name prefix and Field name suffix are applied when generating all field names except record buffer and format buffer.
See Examples for the Generation of FORTRAN Field Names.
- If a resulting field name is not unique within the entire file structure, an error message is given.

FORTRAN Names for Multiple Fields - MU - Contained in Periodic Groups - PE

Before each field of type MC or PC, a counter field is generated, with a name created from the name of the MC or PC field prefixed by C.

For example, a Predict field object DAY with no field name synonym for FORTRAN and with type MC will generate the field names CDAY and DAY.

FORTRAN Names for Additionally Generated Indicator Fields

An additional indicator field prefixed with S is generated in the following cases:

- a field is defined with Suppression option set to U (null allowed)
- a field of a file of type A, U or B is defined with Suppression option set to R (not null)

Note:

For Adabas fields, the additional indicator field is only generated for the following Adabas versions:

- I3 or above
- U1 or above
- V4 or above.

Examples for the Generation of FORTRAN Field Names

Example: with the parameter settings

Field prefix = PR
Field suffix = SUF
Validate = *
Truncation = R

the following field names are generated for a field named "day" of type MC:

PRCDAY, PRDAYS for IBM or Siemens compiler

PRCDAYSUF, PRDAYSUF for VAX/VMS compiler.

Field Format and FORTRAN Copy Code

Fields in the FORTRAN copy code have a FORTRAN clause determined by the following factors:

- length and format of the corresponding Predict field object
- file type of the file containing the field
- the compiler for which you are generating.

PRD Format	PRD Length	File Type or Compiler	FORTRAN Clause	Byte Length
B/I	1	Compiler: Siemens	INTEGER*1	
		Compiler: VMS	BYTE	
		Compiler: IBM, FORTRAN 77	CHARACTER*1	
B	$l=3, 5, 6, 7$		CHARACTER* l	
B/I	2	File Type: BT, BV or Compiler: IBM, Siemens, VMS	INTEGER*2	
		Compiler: FORTRAN 77 and File Type: not BT, BV	CHARACTER*2	
	4	File Type: BT, BV or Compiler: IBM, Siemens, VMS	INTEGER*4	
		Compiler: FORTRAN 77 and File Type: not BT, BV	INTEGER	
	8	IBM, VMS, FORTRAN 77	CHARACTER*8	
		Compiler: Siemens	INTEGER*8	
B	$l \Rightarrow 9$		CHARACTER* l	
F	4	Compiler: IBM, Siemens, VMS	REAL*4	
		Compiler: FORTRAN 77	REAL	
	8	Compiler: IBM, Siemens, VMS	REAL*8	
		Compiler: FORTRAN 77	DOUBLE PRECISION	
N/U NS/US	nn.m		CHARACTER* l	$l=nn+m$
P/PS	nn.m		CHARACTER* l	$l=(nn+m+2)/2$
D		File Type: D, E	CHARACTER*10	
		File Type: BT, BV	CHARACTER*8	
		other file types	CHARACTER*4	
T		File Type: D, E, BT, BV	CHARACTER*8	8
		other file types	CHARACTER*7	7
L		Compiler: IBM, Siemens, VMS	LOGICAL*1	1
		Compiler: FORTRAN 77	CHARACTER*1	1
BT		File Type: YT, YV	see format B, length 1	1

DT		File Type: JT, JV	CHARACTER*25	25
		File Type: OT, OV	CHARACTER*9	9
		File Type: YT, YV	CHARACTER*32	32
DS		File Type: YT, YV	CHARACTER*28	28
G	n	File Type: D, E	CHARACTER*m	m=2*n
GV	n	File Type: D, E	CHARACTER*m	m=2*n
GL	n	File Type: D, E	CHARACTER*m	m=2*n
TS		File Type: D, E	CHARACTER*26	26
		File Type: YT, YV	CHARACTER*8	8
MO		File Type: YT, YV, JT, JV	see format F, length 8	8
MS		File Type: YT, YV	see format F, length 4	4
OK		File Type: JT, JV	CHARACTER*16	16
TK		File Type: JT, JV	CHARACTER*8	8

- If generated for IBM, Siemens or VMS compilers:
Any file number field, length fields and automatically generated counter fields have the clause INTEGER*2.
- If generated for a FORTRAN 77 compiler:
Constants for the file number and length fields have the clause INTEGER. Automatically generated counter fields have the clause CHARACTER*2.

A numeric or binary format field with a length not included in the table above is treated in FORTRAN copy code as an alphanumeric format field and cannot have an initial value generated for it. A warning message is given.

Fields other than redefinitions are always aligned, both in the record buffer and in the format buffer, on the boundary shown in the table above.

The array that is created to show the offset of each item in the file (see the parameter Offset name) is aligned so that its first element, (Offset name(1)), corresponds to offset 0. Similarly, the element number in each EQUIVALENCE statement generated is one more than the offset of the corresponding field. For example, the statement EQUIVALENCE(Offset name(15),elem) in FORTRAN copy code would correspond to a field named elem with offset 14.

In FORTRAN copy code, redefinitions are generated both for elements within a group and for single fields on level 1. Redefinitions are not generated for periodic groups; instead, a warning message is issued.

Note:

If FORTRAN copy code for DB2 tables/views is generated, the format NS or US is converted to the FORTRAN format CHARACTER. Any redefinitions that exist for this field are skipped.

A DB2 large object is generated as locator like an I4 field.

Sample Output

```

13:28:09          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate FORTRAN Copy Code -                Page: 1

File ID .. GENERATION-EXAMPLE

C*****
C   THIS RECORD-BUFFER LAYOUT WAS GENERATED BY PREDICT
C   FOR FILE: GENERATION-EXAMPLE
C   ON: 2003-05-31 STARTING AT 13:28:09
C   FILE-COMMENTS: Example file for the
C   PREDICT generation subsystem.
C   ..
C*****
      CHARACTER*648          GENERA
C >>> RB NAME :
C >>> DIC1809 WARNING: FIELDNAME(S) TRUNCATED
      CHARACTER*1           OFF(648)
      EQUIVALENCE(OFF(1),GENERA)
      CHARACTER*99          GROUP1
      EQUIVALENCE(OFF(1),GROUP1)
      CHARACTER*14          ELN9V5
      EQUIVALENCE(OFF(1),ELN9V5)
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' N ' --> ' A ' )
      CHARACTER*30          GRINGR
      EQUIVALENCE(OFF(15),GRINGR)
C >>> GR-IN-GROUP :
C >>> DIC1809 WARNING: FIELDNAME(S) TRUNCATED
      INTEGER*4             ELEB4
      EQUIVALENCE(OFF(17),ELEB4)
      CHARACTER*4           ELPS52
      EQUIVALENCE(OFF(21),ELPS52)
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ' )
      INTEGER*4             MUB4(5)
      EQUIVALENCE(OFF(25),MUB4)
      CHARACTER*7           MURB7
      EQUIVALENCE(OFF(25),MURB7)
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' )
      CHARACTER*13          MURB13
      EQUIVALENCE(OFF(32),MURB13)

```

```
13:28:09          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate FORTRAN Copy Code -                Page: 2

File ID .. GENERATION-EXAMPLE

C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' NS ' --> ' A ')
      CHARACTER*8          MURB8
      EQUIVALENCE(OFF(25),MURB8)
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ')
      CHARACTER*42         ELEA42
      EQUIVALENCE(OFF(45),ELEA42)
      REAL*8              ELEF8
      EQUIVALENCE(OFF(89),ELEF8)
      CHARACTER*3          FOB3
      EQUIVALENCE(OFF(97),FOB3)
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ')
      INTEGER*2           CPCOCC
      EQUIVALENCE(OFF(101),CPCOCC)
C >>> PC-OCC-7 :
C >>> DIC1809 WARNING: FIELDNAME(S) TRUNCATED
      CHARACTER*534        PCOCC7
      EQUIVALENCE(OFF(103),PCOCC7)
C >>> DIC1848 MESSAGE: PE-GROUP(S) 'STRUCT' DEFINITION SET TO 'N'
      CHARACTER*10         PCELED(7)
      EQUIVALENCE(OFF(103),PCELED)
C >>> PC-ELE-DE-NS-7V3 :
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' NS ' --> ' A ')
C >>> DIC1809 WARNING: FIELDNAME(S) TRUNCATED
      INTEGER*2           CMCP(7)
      EQUIVALENCE(OFF(173),CMCP)
      CHARACTER*4         MCPS(11,7)
      EQUIVALENCE(OFF(187),MCPS)
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ')
      CHARACTER*112        PCGR
      EQUIVALENCE(OFF(495),PCGR)
      INTEGER*2           FOI2(7)
      EQUIVALENCE(OFF(495),FOI2)
      CHARACTER*14         PS20V7(7)
      EQUIVALENCE(OFF(509),PS20V7)
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ')
      REAL*4              FOF4(7)
      EQUIVALENCE(OFF(609),FOF4)
      CHARACTER*4         ELED
      EQUIVALENCE(OFF(637),ELED)
```

```

13:28:09          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate FORTRAN Copy Code -                Page:   3

File ID .. GENERATION-EXAMPLE

C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' D ' --> ' A ' )
      CHARACTER*7          ELET
      EQUIVALENCE(OFF(641),ELET)
C >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' T ' --> ' A ' )
      LOGICAL*1           ELEL
      EQUIVALENCE(OFF(648),ELEL)

C*****
C   THIS FORMAT-BUFFER WAS GENERATED BY PREDICT
C   FOR FILE: GENERATION-EXAMPLE
C   ON: 2003-05-31 STARTING AT 13:28:09
C*****
      CHARACTER*136       FGENER
C >>> FORMAT BUFFER NAME :
C >>> DIC1809 WARNING: FIELDNAME(S) TRUNCATED
      CHARACTER*1         OFFAA(136)
      CHARACTER*60        OFFAB
      1 / 'AB,2X,AD,AE,AF1-5,AG,2X,AH,AI,1X,ALC,2,AM1-7,AN1C,2,AN2C,2,A' /
      CHARACTER*60        OFFAC
      1 / 'N3C,2,AN4C,2,AN5C,2,AN6C,2,AN7C,2,AN1-7(1-11),AP1-7,AQ1-7,2X, /
      CHARACTER*16        OFFAD
      1 / ',AR1-7,AS,AT,AU.' /
      EQUIVALENCE(OFFAA(1),FGENER)
      EQUIVALENCE(OFFAA(1),OFFAB)
      EQUIVALENCE(OFFAA(61),OFFAC)
      EQUIVALENCE(OFFAA(121),OFFAD)
C*****

DIC1800 SUMMARY:      25 FIELD(S) PROCESSED
DIC1809 WARNING:      5 FIELDNAME(S) TRUNCATED
DIC1818 WARNING:     11 FORMAT(S) CHANGED
DIC1819 MESSAGE:      7 SLACK BYTE(S) GENERATED
DIC1847 MESSAGE:      3 FIELD(S) SKIPPED FOR RECORDBUFFER STRUCTURE
DIC1848 MESSAGE:      1 PE-GROUP(S) 'STRUCT' DEFINITION SET TO 'N'

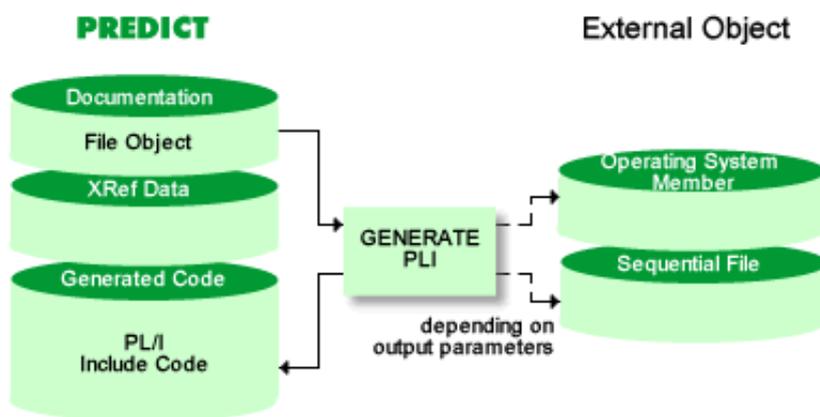
```

PL/I Include Code

The function Generate PL/I Include Code generates a record buffer for use in PL/I programs based on a Predict file object.

In addition, an Adabas format buffer can also be generated if required for files of the following types:

- Adabas file (file type A) with parameter Adabas SQL usage = N
- Adabas userview (file type U)



This section contains:

- Calling the Function
- Generate PL/I Include Code in Batch Mode
- Names in PL/I Include Code
- Field Format and PL/I Include Code
- Sample Output

Calling the Function

The Generate PL/I Include Code screen is displayed with function code G and object code PL in a Predict main menu, or with the command GENERATE PLI.

```

10:13:10          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan   0          - Generate PLI Copy Code -

File ID .....*
Save as member .....          Save in library .... PLILIB
Overwrite option ..... Y (Y,N) Op. system member ..
Punch / output .....* N
List generated code ..... Y (Y,N) List offsets .....* N
Generate format buffer ....* N Adabas version ....* I7
Check field name .....* A Field name prefix .. ADABAS_
Start level ..... 1 (0-40) Field name suffix ..
Level number increment ..... 1 (1-40) Validate ..... _
Level shift increment ..... 3 (0-9) Truncation .....* R
Nr. of abstract lines ..... 4 (0-16) Align .....* N
Generate initial value ....* N Static ..... N (Y,N)
Structure as char ..... N (Y,N) Numeric sign .....* T
With DCL ..... N (Y,N) Position of sign ..* R
Record buffer name .....
Format buffer name .....

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
    
```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presettings	
The parameters below can be changed in the Modify PL/I Defaults screen. See Generation Defaults.	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated. See Parameters Specifying the Form of Output for more information.
Parameters	
The parameters Save as member, Save in library, Overwrite option, Op. system member, Punch/output, Workfile name and List generated code are described in the section Parameters Specifying the Form of Output.	
File ID	ID of the Predict file object from which the definitions are to be generated. Enter an asterisk to display a selection screen. Alternatively, use an asterisk as a wildcard. For example, enter A* to display IDs beginning with A only.

List offsets	<p>Y The offset of each item in the record buffer structure (relative to the beginning of the structure) in decimal and hexadecimal formats is to be included as a comment. The total length of each buffer is also included.</p> <p>P As above, but the absolute position (offset+1) is included as a comment.</p> <p>L The total lengths of the record buffer and the format buffer are to be included as a comment.</p> <p>V The file number and the calculated lengths of the record buffer and the format buffer are to be generated as constants in the include code. The name of the file number constant will be the record buffer name prefixed by N_. The name of each length constant will be the appropriate buffer name prefixed by L_. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.</p> <p>N No offset.</p>
Generate format buffer	<p>The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas SQL usage set to N) or for files of type U. Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, you must set this parameter to F or N.</p>
Adabas version	<p>The version of Adabas for which the include code is to be generated. Enter an asterisk for valid values or see table in the section Adabas Version for more information.</p>
Check field names	<p>A The field names must be unique throughout the entire structure</p> <p>Y Structure levels are included in the validation check of the field names: if two fields have the same name, they must be separated by at least one field with a different name and a lower level number.</p> <p>N The field names in the generated PL/I code will not be checked for uniqueness.</p>
Start level	<p>The starting level number to be used for the generated statements.</p> <p>Note: For fields within a redefinition, Start level is always 1.</p>

Field name prefix	The prefix appended to each field name generated.												
Field name suffix	The suffix appended to each field name generated.												
Level number increment	<p>The increment to be used when assigning level numbers to the generated statements. This parameter is evaluated in conjunction with parameter Start level. See table below.</p> <p>Note: This parameter is ignored for fields within a redefinition.</p> <table border="1"> <thead> <tr> <th>Start Level</th> <th>Level number increment (Example)</th> <th>Level Numbers</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>5</td> <td>1, 6, 11...</td> </tr> <tr> <td>1</td> <td>5</td> <td>1, 5, 10 ...</td> </tr> <tr> <td>2 - 40</td> <td>5</td> <td>n, n+5, n+10...</td> </tr> </tbody> </table>	Start Level	Level number increment (Example)	Level Numbers	0	5	1, 6, 11...	1	5	1, 5, 10 ...	2 - 40	5	n, n+5, n+10...
Start Level	Level number increment (Example)	Level Numbers											
0	5	1, 6, 11...											
1	5	1, 5, 10 ...											
2 - 40	5	n, n+5, n+10...											
Level shift increment	<p>The number of positions to be shifted right when a level number which is higher than the current level number is encountered.</p> <p>Note: This parameter is ignored for fields within a redefinition.</p>												
Validate	<p>Determines how invalid characters are handled.</p> <p>blank Invalid characters in a field name will result in an error message but will not be modified.</p> <p>rep.char Invalid characters in a field name are replaced by this character. Valid replace characters: letters A-Z, digits 0-9, \$, %, # and _ (underscore).</p> <p>*</p> <p>Invalid characters in a field name are deleted.</p>												
Nr. of abstract lines	The number of Predict abstract lines per field to be included in the generated code.												
Truncation	<p>Specifies which characters are deleted if a generated field name is too long:</p> <p>L from the left</p> <p>R from the right</p> <p>M from the middle</p> <p>A warning is given if field names are truncated.</p>												

Generate initial value	<p>Y Statements are generated to initialize the structure with the value for Init value defined for the field objects in Predict. Fields with no value defined for Init value are initialized with zeros or spaces.</p> <p>S Only fields which have a value defined for Init value in the corresponding Predict object are initialized.</p> <p>N No initialization.</p>
Align	<p>Determines which fields are to be aligned. In the record buffer and in the format buffer fields are aligned on the boundary shown in the table in the section Field Format and PL/I Include Code.</p> <p>Note: This parameter only takes effect with fields that have the PL/I attribute FIXED BIN or FLOAT DEC.</p> <p>Y All fields are aligned.</p> <p>N No fields will be aligned. Predict will add the PL/I keyword UNAL to all appropriate fields.</p> <p>S Fields are aligned only if the 3GL specification Synchronized of the corresponding Predict object is set to S. Otherwise Predict adds the PL/I keyword UNAL.</p>
Static	The structure is declared with the attribute STATIC .
Structure as char	The entire generated structure is declared at the end of the record buffer as a single character string.
Numeric sign	Specifies which of the PL/I picture characters T, I or R is to be used for the representation of numeric values of format packed with sign or unpacked with sign.
With DCL	Y The generated code is preceded by a declare statement DCL and ends with a semicolon (not a comma). PL/I copy code can be inserted in any position of a PL/I program, it does not have to be inserted at the beginning.
Position of sign	Defines the position of the sign in a numeric field R right L left.
Record buffer name	Specifies the name of the record buffer in the generated structure. If omitted, the file ID is used.
Format buffer name	Specifies the name of the format buffer in the generated structure. If omitted, the file ID prefixed by FORBUF_ is used.

Generate PL/I Include Code in Batch Mode

Command: GENERATE PLI

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional. If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Save as member	MEM	2
Save in library	LIB	3
Overwrite option	REPLACE	4
Op. system member	OS-MEMBER	5
Start level	START-LEVEL	6
Field name prefix	PREFIX	7
Level number increment	LEVEL-INCREMENT	8
Field name suffix	SUFFIX	9
Level shift increment	LEVEL-SHIFT	10
Nr. of abstract lines	NR	11
Validate	VALIDATION	12
Generate initial value	INIT	13
Truncation	TRUNCATION	14
Generate format buffer	FORMAT-BUFFER	15
List offsets	OFFSET	16
Structure as char	STRUCTURE	17
List generated code	LIST	18
Static	STATIC	19
Punch / output	PUNCH	20
Record buffer name	RECORD-BUFFER-NAME	21
Format buffer name	FORMAT-BUFFER-NAME	22
Align	SYNC	23
With DCL	DCL	24
Adabas version	ADA-VER	25
Check field name	CHECK-NAME	26
Numeric sign	NUM-SIGN	27
Position of sign	POS-SIGN	28
Workfile name (see note below)	WORKFILE-NAME	29
If Entire System Server is used		

- DB-ID	NP-DBID	30
- Dataset	NP-DSNAME	31
- Volume	NP-VOLSER	32
- Library	NP-LIB	33
- Sublibrary	NP-SUBLIB	34
- Member type	NP-MEMTYPE	35
- VSAM catalog	NP-VSAMCAT	36

Note:

You can not specify a value other than blank (' ') for WORKFILE-NAME on mainframes.

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

Names in PL/I Include Code

The following rules apply to the generation of names for PL/I copy code.

- Field names are derived from Predict field objects: either the Field Name Synonym for PL/I or, if none exists, the Predict field ID.
- Alphabetic characters (letters) in Predict object IDs, such as file and field IDs, are converted to upper-case.
- All field names are prefixed, suffixed, validated and if necessary truncated to 31 characters according to the setting of these four options.

For example, with Field name prefix set to Adabas/, Field name suffix to *PERSONNELOFFICE, Validate to _ and Truncation to R the following names are generated for a field named "day" of type MC.

ADABAS_C_HOURS_DAY_PERSONNELOFF
ADABAS_HOURS_DAY_PERSONNELOFFIC

- If a resulting field name is not unique, an error message may be issued if Check field name is either set to A or Y.
- The record buffer name and the format buffer name do not receive the prefix and suffix. The value(s) supplied must not contain invalid characters. If the file ID is taken as default record buffer name or, with the prefix FORBUF_, as default format buffer name, it is validated and if necessary truncated.

PL/I Names for Counter Fields - MC or PC

Before each field of type MC or PC, a counter field is generated, with a name created from the name of the MC or PC field prefixed by C_.

For example, a Predict field object HOURS-DAY with no field name synonym for PL/I and with type MC will result in two fields C_HOURS-DAY and HOURS-DAY.

PL/I Names for Additionally Generated Indicator Fields

An additional indicator field prefixed with S_ is generated in the following cases:

- a field is defined with Suppression option set to U (null allowed)
- a field of a file of type A, U or B is defined with Suppression option set to R (not null)

Note:

For Adabas fields, the additional indicator field is only generated for the following Adabas versions:

- I3 or above
- U1 or above
- V4 or above.

Field Format and PL/I Include Code

Fields in the PL/I include code have a PL/I clause determined by the length and format of the corresponding Predict field object, as shown in the table below where s is the numeric sign whose content (T, I, or R) and position (left or right) are defined in the PL/I generation defaults; nn+m must not exceed 15; and if m is zero, V(m)9 is omitted.

Predict Format	File Type	Predict Length	PL/I Clause	Note
A		nnn	CHAR(nnn)	
B/I		1	BIT(8)	
B		3	BIT(24)	
B/I		2	FIXED BIN(15,0)	Aligned at half word boundary (if word length=4)
		4	FIXED BIN(31,0)	Aligned at word boundary (if word length=4)
B		<i>l</i> =>5	CHAR(<i>l</i>)	
I		8	CHAR(8)	
F		4	FLOAT DEC(6)	Aligned at word boundary (if word length=4)
		8	FLOAT DEC(16)	Aligned at double word boundary (if word length=4)
N/U		nn.m (nn+m<16)	PIC '(nn)9V(m)9'	If m = 0, V(m)9 is not applicable.
		nn.m (nn+m>15)	CHAR(nn+m)	
NS/US		nn.m (nn+m<16)	PIC '(nn)9V(m-1)9s' or PIC 's(nn-1)9V(m)9'.	s represents the numeric sign, the contents of which (T, I or R) and Position (left or right) depend on the parameters Numeric sign and Position of sign If m = 0, V(m) 9 is not applicable.
		nn.m (nn+m>15)	CHAR(nn+m)	
P/PS		nn.m (nn+m<16)	FIXED(nn+m,m)	
		nn.m (nn+m>15)	CHAR(<i>l</i>)	
D	D, E		CHAR(10)	
	other		FIXED(6)	
T	D, E		CHAR(8)	
	other		FIXED(12)	

L			BIT(8)	
DT	OT, OV		CHAR(9)	
	JT, JV		CHAR(25)	
G, GV, GL	D, E	n	CHAR(m)	m=2*n
JT, JV	BV	n	CHAR(n)	
	BL	n	CHAR(n)	Field is skipped if no length is specified
MO	JT, JV		FLOAT DEC(16)	
OK	JT, JV		CHAR(16)	
TK	JT, JV		CHAR(8)	

File Type in Predict	Predict Format	Predict Char. Set	Length	PL/I Clause	Note
OT, OV	A, AL	any	n	CHAR(n)	
	AV	there is only one char. set	n	Two-level group is generated: 02 Fieldname 03 Fieldname-LEN FIXED BIN (15,0) 03 Fieldname-TXT CHAR (n)	
JT, JV	A, AV	any	n	CHAR(n)	
	AL	there is only one char. set	n	CHAR(n)	Field is skipped if no length is specified
D, E	A, AL	any	n	CHAR(n)	
	AV	any	n	Two-level group is generated: 02 Fieldname 03 Fieldname-LEN FIXED BIN (15,0) 03 Fieldname-TXT CHAR (n)	
A, U	A	there is only one char. set	n	CHAR(n)	
	AV	there is only one char. set	n	Two-level group is generated: 02 Fieldname 03 Fieldname-LEN FIXED BIN (15,0) 03 Fieldname-TXT CHAR (n)	
A(SQL) AT, B	A, AV	there is only one char. set	n	CHAR(n)	

Additional Notes

An automatically generated counter field has the clause FIXED BIN (15,0).

A numeric or binary format field with a length not included in the table above is treated in PL/I include code as an alphanumeric format field and cannot have an initial value generated for it. A warning message is issued.

See also description of parameter Align.

If PL/I include code for DB2 tables/views is generated, the format NS or US is converted to the PL/I format FIXED. Any redefinitions that exist for this field are skipped.

A DB2 large object is generated as locator like an I4 field.

Sample Output

```

13:28:02          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate PL/I Copy Code -                   Page: 1

File ID .. GENERATION-EXAMPLE

/*****
/*   THIS RECORD-BUFFER LAYOUT WAS GENERATED BY PREDICT          */
/*   FOR FILE: GENERATION-EXAMPLE                                */
/*   ON: 2003-05-31 STARTING AT 13:28:02                        */
/*   FILE-COMMENTS: Example file for the                        */
/*   PREDICT generation subsystem.                               */
/*   ..                                                           */
/*****

01 GENERATION_EXAMPLE,
   02 GROUP_1 ,
       03 ELE_N_9V5          PIC '(9)9V(5)9',
       03 GR_IN_GROUP ,
           04 ELE_B_4          FIXED BIN (31,0),
           04 ELE_PS_5V2       FIXED (7,02),
           04 MU_B_4 (5)       FIXED BIN (31,0),
       03 ELE_A_42          CHAR (42),
       03 ELE_F_8          FLOAT DEC (16),
       03 ELE_B_3          BIT(24) UNAL,
   02 C_PC_OCC_7          FIXED BIN (15,0),
   02 PC_OCC_7 (7) ,
       03 PC_ELE_DE_NS_7V3     PIC '(7)9V(2)9T',
       03 C_PC_MC_PS_6V1       FIXED BIN (15,0),
       03 PC_MC_PS_6V1 (11)     FIXED (7,01),
       03 PC_GR ,
           04 PC_ELE_I_2       FIXED BIN (15,0),
           04 PC_PS_20V7       CHAR (14),
/* >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ') */
       03 PC_ELE_F_4          FLOAT DEC (6),
   02 ELE_D          FIXED (6,00),
   02 ELE_T          FIXED (12,00),
   02 ELE_L          BIT(8),
01 MU_B_4_STRUCT
   BASED(ADDR(GENERATION_EXAMPLE.MU_B_4)),
   02 MU_RED_B_7          CHAR (7),

```

```

13:28:02          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Generate PL/I Copy Code -                   Page: 2

File ID .. GENERATION-EXAMPLE

/* >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' )      */
   02 MU_RED_US_13          PIC '(12)9T',

01 MU_B_4_STRUCT1
   BASED(ADDR(GENERATION_EXAMPLE.MU_B_4)),
   02 MU_RED_B_8          CHAR (8),
/* >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' )      */
/* ++ CONSTANTS FOR RECORD BUFFER LENGTH AND FILE NUMBER          */
01 L_GENERATION_EXAMPLE    FIXED BIN (15,0) UNAL INIT(674),
01 N_GENERATION_EXAMPLE    FIXED BIN (15,0) UNAL INIT(231),
/*****
/* THIS FORMAT-BUFFER WAS GENERATED BY PREDICT                      */
/* FOR FILE: GENERATION-EXAMPLE                                      */
/* ON: 2003-05-31 STARTING AT 13:28:03                              */
/*****
01 FORBUF_GENERATION_EXAMPLE_1 ,
   02 FILLE001 CHAR(34) INIT('AB,2X,AD,AE,AF1-5,AG,2X,AH,AI,1X,A'),
   02 FILLE002 CHAR(34) INIT('LC,2,AM1,AN1C,2,AN1(1-11),AO1,2X,A'),
   02 FILLE003 CHAR(34) INIT('R1,2X,AM2,AN2C,2,AN2(1-11),AO2,2X, '),
   02 FILLE004 CHAR(34) INIT('AR2,2X,AM3,AN3C,2,AN3(1-11),AO3,2X, '),
   02 FILLE005 CHAR(34) INIT(' ,AR3,2X,AM4,AN4C,2,AN4(1-11),AO4,2, '),
   02 FILLE006 CHAR(34) INIT('X,AR4,2X,AM5,AN5C,2,AN5(1-11),AO5, '),
   02 FILLE007 CHAR(34) INIT('2X,AR5,2X,AM6,AN6C,2,AN6(1-11),AO6, '),
   02 FILLE008 CHAR(34) INIT(' ,2X,AR6,2X,AM7,AN7C,2,AN7(1-11),AO, '),
   02 FILLE009 CHAR(21) INIT('7,2X,AR7,2X,AS,AT,AU. '),
01 FORBUF_GENERATION_EXAMPLE CHAR(293)
   BASED (ADDR( FORBUF_GENERATION_EXAMPLE_1 )),
/*++ CONSTANT FOR FORMAT BUFFER LENGTH                              */
01 L_FORBUF_GENERATION_EXAMPLE    FIXED BIN (15,0) UNAL INIT(293),
/*****

DIC1800 SUMMARY:      25 FIELD(S) PROCESSED
DIC1818 WARNING:      3 FORMAT(S) CHANGED
DIC1819 MESSAGE:      33 SLACK BYTE(S) GENERATED
DIC1847 MESSAGE:      3 FIELD(S) SKIPPED FOR RECORDBUFFER STRUCTURE

```

Sample File Definition

The following file definition was used for generating the sample output for the following external object types:

- ADACMP, ADAWAN or ADAFDU definitions
- Adabas files
- ADAINV definitions
- ADASCR definitions
- Vista translator table
- DDM
- Assembler copy code
- C include Code
- COBOL copy code
- FORTRAN copy code
- PL/I include Code

Note:

The file definitions used for generating a DB2 table and an Adabas VSAM Bridge Transparency table are given in the respective sections of this section.

```

13:58:55          ***** P R E D I C T  4.3.1  *****                2003-05-31
                   - Display File with Children -                        Page:    1

File ID ..... * GENERATION-EXAMPLE
Type ..... Adabas, Simple file
File number ..... 231
                                     Added 2003-05-31 at 13:30 by BOE
                                     Modified 2003-05-31 at 13:35 by HNO
                                     Fields modified 2003-05-31 at 13:37 by CHD
-----

File attributes
Sequence field .....
Log. distribution type .. Simple file
Adabas SQL usage ..... N

Vista
L-DBnr ..... 3
L-Fnr ..... 241

Abstract
Example file for the
Predict generation subsystem.
..

```

```

13:58:55          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Display File with Children -                Page: 2

File ID ..... * GENERATION-EXAMPLE
-----

Adabas attributes
Required attributes
  Phys. file number ..... 231          Added 2003-05-31 at 13:30 by BOE
  Min ISN ..... 1                    Modified 2003-05-31 at 13:35 by CHD
  Max ISN ..... 3391

Loading attributes
  Max recl .....
  ISN reusage ..... N
  User ISN ..... N
  Ciphared ..... N
  LOWNERID .....
  Refresh from program .. N
  Automatic allocation .. Y
  PLOG ..... Y
  ISN SIZE ..... 0 Adabas decides

Loading attributes
  One AC extent .... N
  DS reusage ..... Y
  Mixed DS device .. N
  Mirror ..... N

Adabas security definition
  Access level ... 2
  Update level ... 3

Extend allocation
  Device   Padding factor   Cylinder   Blocks   Max 2. Alloc
  Asso 3390           10           UI         41
                   NI         80
  Data 3390           10           DS   > 1   233
    
```

```

13:58:55          ***** P R E D I C T 4.3.1 *****          2003-05-31
                   - Display File with Children -                Page:   3

File ID ..... * GENERATION-EXAMPLE
-----
                   ----- Generation -----
Generation status      on      DBnr      Fnr      Warnings
NATURAL DDM GENERATED 2003-05-31  180      171      NATURAL DDM GENERATED
VERIFIC. RULE GENERATED 2003-05-31  180      171      VERIFIC. RULE GENERATED

Cnt  Ty L Field ID          F      Length  Occ D DB S
-----
  1 GR 1 GROUP-1              AA
  2  2 ELE-N-9V5              N      9.5      AB N
      Specifications for 3GL
      Initialize with .... Z      Zero
      Field synonyms
      BAL/Assembler ..... ELN9V5
      FORTRAN ..... ELN9V5
      Verification name
      GEN-CHECK-ZERO
  3 GR 2 GR-IN-GROUP          AC
      Field synonyms
      BAL/Assembler ..... GR-IN-GR
  4  3 ELE-B-4                B      4.0      D AD F
      Specifications for 3GL
      Init value ..... 00001111
      Condition name
      INIT-ELE-B-4
      : Condition value
      : 00001000
      COND-ELE-B-4-V-1
      : Condition value
      : FFFFFFFF
      Verification name
      GEN-VER-EXAM
  5  3 ELE-PS-5V2            PS      5.2      AE N
      Specifications for 3GL
      Initialize with .... Z      Zero
      Field synonyms
      BAL/Assembler ..... ELPS52
      FORTRAN ..... ELPS52
      Verification name
      GEN-CHECK-ZERO
    
```

```

13:58:55          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Display File with Children -                Page: 4

File ID ..... * GENERATION-EXAMPLE
-----
 6 MU 3 MU-B-4          B          4.0          5  AF N
 7 RE 3 MU-B-4
 8   4 MU-RED-B-7      B          7.0
    Field synonyms
    BAL/Assembler ..... MURB7
    FORTRAN ..... MURB7
 9   4 MU-RED-US-13    US         13.0
    Field synonyms
    BAL/Assembler ..... MURB13
    FORTRAN ..... MURB13

10 RE 3 MU-B-4
11   4 MU-RED-B-8      B          8.0
    Condition name
    MU-RED-B-8-COND1
    : Condition value
    : 111111
    Field synonyms
    BAL/Assembler ..... MURB8
    FORTRAN ..... MURB8
12   2 ELE-A-42        A         42.0          AG N
    Specifications for 3GL
    Initialize with .... H    High-value
    Condition name
    COND-NAME-START
    : Condition value
    : COND-START
    : THRU
    : COND-ZEND
    FILL-WITH-STRING
    : Condition value
    : A
    Verification name
    GEN-CHECK-BLANK
    GEN-VER-EXAM
13   2 ELE-F-8          F          8.0          AH F
14   2 ELE-B-3          B          3.0          AI N
    Field synonyms
    FORTRAN ..... FOB3
    
```

13:58:55	***** P R E D I C T 4.3.1 *****				2003-05-31
	- Display File with Children -				Page: 5
File ID * GENERATION-EXAMPLE					

15	SB 1	SB-ELE-A-5	A	5.0	AJ N
		Source field(s)	Start	End	DB
		ELE-A-42	10	14	AG(10-14)
		Field synonyms			
		BAL/Assembler			SBB5
		FORTRAN			SBB5
16	SP 1	SP-DE-A-24	A	24.0	D S1 N
		Source field(s)	Start	End	DB
		ELE-B-3			AI(1-3)
		ELE-A-42	20	36	AG(20-36)
		ELE-PS-5V2			AE(1-4)
		Field synonyms			
		BAL/Assembler			SPDA24
		FORTRAN			SPDA24
17	PC 1	PC-OCC-7			7 AL
18	2	PC-ELE-DE-NS-7V3	NS	7.3	D AM N
		Field synonyms			
		BAL/Assembler			PCNS7V3
19	MC 2	PC-MC-PS-6V1	PS	6.1	11 D AN N
		Field synonyms			
		BAL/Assembler			PCMCPS
		FORTRAN			MCPS
		Verification name			
		GEN-CHECK-ZERO			
20	GR 2	PC-GR			AO
21	3	PC-ELE-I-2	I	2.0	AP F
		Field synonyms			
		BAL/Assembler			BALI2
		FORTRAN			FOI2
22	3	PC-PS-20V7	PS	20.7	AQ N
		Field synonyms			
		BAL/Assembler			PCPS20V7
		FORTRAN			PS20V7
23	2	PC-ELE-F-4	F	4.0	AR F
		Field synonyms			
		BAL/Assembler			BALF4
		FORTRAN			FOF4
24	1	ELE-D	D		AS N
25	1	ELE-T	T		AT N
26	1	ELE-L	L		AU N
27	HQ 1	HQ-DE-I-4	I	4.0	D S2
		Source field(s)			DB
		PC-ELE-I-2			AP
		MU-B-4			AF
*** End of report ***					

File Implementation

When generating external objects from Predict file objects, several generation tasks can be executed comfortably in one run using an implementation plan.

An implementation plan is defined online and can be executed either online or in batch mode.

How this section is Organized

This section covers the following topics:

- Concepts of File Implementation
- Calling File Implementation Functions
- File Implementation Functions

The individual generation functions are described in the section Generation in this documentation.

Concepts of File Implementation

Generation functions can be executed from an implementation plan.

Each generation task contained in an implementation plan calls a generation function once.

A generation task comprises the following:

- the call of the generation function (for example GENERATE FDT)
- the Predict file object to be processed
- a complete set of generation options to be used for generation
- status information.

Implementation plans can be created, extended, copied, displayed, modified, renamed, purged and executed. An implementation plan to be processed can be selected from a list.

Benefits of Implementation Plans

Whenever many generation operations are to be performed repeatedly, it is advisable to create an implementation plan containing the generation tasks. Here are some examples of where to use an implementation plan:

- Creating DDMs for a Natural application. If all files to be used in an information processing system are defined in Predict, all DDMs to be used by Natural programs can easily be generated (and regenerated) using an implementation plan.
- Creating all copy code used in an application written in a third generation language.
- Implementing Vista elements for complex data distribution structures. If data is distributed across several machines in a complex (possibly heterogeneous) network, implementing the required Vista elements may require many calls of the generation function GENERATE VISTATAB. These calls can be executed comfortably from an implementation plan.

Note:

Using an implementation plan not only helps when several generation functions have to be called one after another, but also helps keep track of data structures of an application or of distributed data structures.

Overview of Activities

The following activities can or must be performed when using an implementation plan:

Create an Implementation Plan

Implementation plans are created with the Add implementation plan function by specifying a Plan ID in the File Implementation menu and - in the subsequent Add implementation plan screen - a combination of Predict selection criteria and generation function(s).

Extend an Implementation Plan

The function Extend implementation plan is used to add generation tasks for file objects that are not yet contained in the plan.

Modify an Implementation Plan

The function Modify implementation plan is used for various tasks:

- Add generation tasks for file objects that are already contained in a plan (command IN).
- Display external object (command DI).
- Specify which generation options are to be used (commands OO, MO, DO).
- Display messages of generation functions called from the plan (command SM).
- Determine that a generation task is to be executed again (command RE).
- Delete generation tasks from plan (command UN).

Execute an Implementation Plan

The function Execute implementation plan processes all generation tasks with status no or reexecute.

Depending on the result of a generation function, the generation status of the tasks is changed. All messages of the generation function are stored and can be displayed with command SM of the function Modify implementation plan.

Reexecute an Implementation Plan

The function Reexecute implementation plan changes the status of all tasks which could be reset to status reexecute manually, to reexecute and then starts the execution of the implementation plan.

Administrate an Implementation Plan

Implementation plans can be renamed, copied, displayed, purged, and selected from a list.

Executing Implementation Plans under Adabas Vista

If an implementation plan contains a generation task for an Adabas file, ADAINV cards, a DDM or Vista translation tables, the virtual machine and the network that were current when adding the task(s) to the plan are stored with the plan.

Such an implementation plan can be modified/extended/executed only if the current virtual machine/network and the virtual machine/network that are stored with the plan are identical.

Generating a DDM under Natural Security

If you use an implementation plan to generate a DDM that is protected by Natural Security, the task is set to mis (valid generation options missing) because a countersignature is required. The function MO (modify generation options) must be performed and a countersignature entered.

If the Predict file object and the Natural Security definition have not been modified since countersigning the task, the generate DDM function will then be performed when the implementation plan is executed.

If either the Predict file object or the Natural Security definition has been modified since countersigning, the generation task will be skipped.

Calling File Implementation Functions

The File Implementation Menu

File implementation functions are called from the File Implementation Menu, which is called with function code F from any Predict main menu or with the command IMPLEMENT FILE.

```

13:13:01          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan 10           - File Implementation Menu -                    Profile HNO

Function                                     Function

A Add an implementation plan                 D Display implementation plan
C Copy implementation plan                   E Extend implementation plan
M Modify implementation plan                 S Select implementation plan from list
N Rename implementation plan                 X Execute implementation plan
P Purge implementation plan                 R Reexecute implementation plan

Function .....

Implementation plan ID ...
Copy ID .....

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJl Flip Print Impl AdmFi SelFi Prof Main
    
```

Parameters	
Function	Select function code from menu.
Implementation Plan ID	ID of implementation plan to be processed. Up to 12 alphanumeric characters and starting with a letter.
Copy ID	For Copy implementation plan function: ID of new implementation plan to be created. Up to 12 alphanumeric characters and starting with a letter.

Executing and Displaying an Implementation Plan in Batch Mode

Implementation plans can be executed, reexecuted and displayed in batch mode.

Enter command IMPLEMENT FILE, and on the next line the required parameters in positional or keyword form.

Field	Keyword	Position
Function	FUNCTION	1
Plan ID	PLAN-ID	2

Example_Batch

To execute and list the implementation plan IMP-TEST use the commands:

```
IMPLEMENT FILE
FUNCTION=X, PLAN-ID=IMP-TEST
IMPLEMENT FILE
D, IMP-TEST
```

Handling of Errors in Batch Mode

Condition Code 106 is issued if

- a function in the implementation plan is executed with errors,
- or a function terminates abnormally,
- or the plan contains at least one generation task that failed or could not be executed (status impossible).

File Implementation Functions

Add Implementation Plan - Code A

Implementation plans are created with the Add implementation plan function by specifying an Implementation Plan ID in the File implementation Menu and - in the subsequent Add Implementation Plan screen - a combination of Predict selection criteria and generation functions.

Invalid combinations of file object and generation function are not added to the implementation plan.

Generation tasks are added for each file that meets the given selection criteria according to the following rules:

- for each valid combination of file object and generation function, one generation task is added (if no external object for the file exists)
- if external objects for a file already exist, one generation task is added for each external object. This facilitates regenerating existing external objects.

Note:

For Adabas files, DDMs, ADAINV cards and Vista translation tables, tasks are added only for those external objects that are accessible in the current virtual machine/network.

The plan is displayed and the tasks can be modified with the same functions that are provided in the Modify Implementation Plan screen

```

13:13:17          ***** P R E D I C T 4.3.1 *****          2003-05-31
          - Add Implementation Plan -          Profile HNO
Implementation plan ID .. HNO-PLAN1          Added 2003-05-31 at 13:12 by HNO
                                          Modified
                                          Executed

File ID .....
Files of type ....*      File number ...
Contained in DA ...

Implemented ..... N (Y,N)
Modified ..... N (Y,N)
with userviews .... N (Y,N)  Restrictions .....*  Profile HNO ,used

----- Generation functions -----
ADACMP/ADAWAN ..... Adabas file ..... ADAINV cards .....
ADASCR ..... Adabas - VSAM ..... DB2 table/view .....
SQL CREATE Stmt ..... DDM for Natural ..... OS/400 File Def. ....
BAL/ASSEMBLER ..... Language C ..... COBOL .....
FORTRAN ..... PL/1 .....

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Next  Stop  Last  LnkEl  Flip  Print  Impl  AdmFi  SelFi  Prof  Main

```

Parameters	
Implementation plan ID	ID of the plan. A read-only field.
Added	Date and time the plan was added.
Modified	Date and time the plan was last modified.
Executed	For function Extend implementation plan: Date and time the plan was executed last. Date and time are taken when executing the last task in the plan. Not applicable for function Add implementation plan.
File ID	ID of the file(s) for which generation tasks are to be included in the plan. Use asterisk notation to select a range of files. If a unique file ID is specified, all other selection criteria are ignored.
Files of type	Only files of this type are added to the plan.
File number	Only files with this number are added to the plan.
Contained in DA	Only files in the specified database(s) are added to the plan.
Implemented	Y A generation task will be included in the plan only for Predict file objects for which implemented external objects exist for the respective generation function. Each generation task will therefore replace an external object.
Modified	Y A generation task will be included in the plan only for Predict objects marked modified after generation. Each generation task will therefore replace an external object if the Predict object has been changed since generation.
with userviews	Y The selection criteria will apply to master files only. If userviews are derived from the Predict file objects, generation tasks for these userviews are automatically included into the implementation plan. For files of type AT (Adabas cluster table), only one file is entered in the plan. The generation task automatically generates a cluster description for each file in the cluster.
Restrictions	See Restrictions in the section Predict User Interface in the Introduction to Predict documentation .
Generation functions	Mark with any non-blank character the external object(s) to be created for each file selected.

Copy Implementation Plan - Code C

Copies the plan Implementation Plan ID to Copy ID.
The IDs are entered in the File Implementation Menu.

Modify Implementation Plan - Code M

Generation tasks in an implementation plan can be changed with several functions. If a task is modified, its generation status (indicated in the column Generation) is changed to reexecute if it had already been executed.

Note:

Plans containing a generation task for an Adabas file, ADAINV cards, a DDM or Vista translation tables can only be modified if the current virtual machine/network and the virtual machine/network that are stored with the plan are identical. It is however possible to purge a plan regardless of the current virtual machine/network.

F	<p>Two-letter command processing the generation task.</p> <p>IN Insert new task for the file object with the same language. The task is inserted after current task. Different generation options can be used.</p> <p>DI Display external object.</p> <p>DO Use default options. The external object will be generated with the options defined in the Modify ... Defaults screen.</p> <p>OO Use old generation options for generation.</p> <p>MO Modify generation options. The respective generation screen is displayed. File ID, Overwrite option and List generated code cannot be modified and Member and Library cannot be changed if the external object already exists.</p> <p>SM Show messages that the generation function issued when the generation task was executed last.</p> <p>RE Reexecute a generation task that was executed successfully or which terminated abnormally.</p> <p>UN Undo generation task (delete from list).</p> <p>Note: Which functions can be applied to a generation task depends on several factors, such as type of generation function and whether the generation task has already been executed. Enter an asterisk in the column F to show valid functions for a specific generation task.</p>
Language	The type of external object to be generated, for example: COBOL when generating COBOL copy code.
Dbnr / Fnr	Database number / file number of the external object (if applicable).
Library / Member	Member and library of the external object (if applicable).
Op.-mem.	See description of parameters Save as member, Save in library and Op. sys member in the section Parameters Specifying the Form of Output.
Opt	<p>Shows which generation options are used:</p> <p>def Execute task using default generation options. See the section Defaults in the Predict Administration documentation.</p> <p>mis No valid generation options are available. The function MO (modify generation options) must be performed for the generation task before it can be executed.</p> <p>old Execute task using generation options that were used when the member was generated last.</p> <p>mod Execute task using modified generation options. This value appears if the command MO has been issued for a generation task (even if no options were in fact changed).</p>

Generation	Status of the generation tasks:
abended	Task terminated abnormally.
errorTask	performed with errors. See also Error Handling - Online.
executing	Task is executing or terminated abnormally.
impossible	Task could not be performed, for example because the generation options are missing or because you are not authorized to generate the DDM. This status can be reset with the command MO (modify generation options). See above.
interrupted	Task interrupted by the user.
no	Task not yet performed.
performed	Task performed successfully.
reexecute	Task will be performed again.
sav.error	Task performed with errors; the generated code was saved.
warning	Task performed with warnings.

Rename Implementation Plan - Code N

After calling the function Rename implementation plan, a screen appears for entering a new Implementation Plan ID.

Purge Implementation Plan - Code P

Additional confirmation is requested before a plan is actually purged. The implementation plan will be purged if you enter DELETE.

Display Implementation Plan - Code D

The function Display implementation plan displays a plan as shown below:

```

13:42:18          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Display Implementation Plan -
                                Added 2003-05-31 at 13:01 by CHD
                                Modified
                                Executed 2003-05-31 at 13:05

Implementation plan ID .. TEST
Network .....
Virtual machine .....
-----
      Language      DBnr   Fnr Library  Member  Op.-mem. Option  Generation
File ID ..... CHD-CIPHER
  Language C
    DIC1800 SUMMARY:    4 FIELD(S) PROCESSED
    DIC1818 WARNING:    1 FORMAT(S) CHANGED
  COBOL
    DIC1800 SUMMARY:    4 FIELD(S) PROCESSED

File ID ..... CHD-COBOL-PREF-G
  Language C
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
          Quit          RFind Flip -   +           Left Right
                                           line 1 of 47 (line 10 on screen)
    
```

Information Given by Display Implementation Plan

The function Display Implementation Plan gives the following information:

- Information as contained in the Add/Extend/Modify Implementation Plan screens.
- If a plan contains a generation task for an Adabas file, ADAINV cards, a DDM or Vista translation tables, the virtual machine/network combination that is stored with the plan is displayed. If the current virtual machine/network combination differs from the one that is stored with the plan, a warning is issued.
- Warnings and error messages issued by generation functions are displayed if the generation function was executed from the plan.

Note:

Display Implementation Plan can be executed in batch mode. See Executing and Displaying an Implementation Plan in Batch Mode.

Extend Implementation Plan - Code E

This function is used to add generation tasks for file objects that are not yet contained in the implementation plan.

The function Extend implementation plan uses the same input screens as Add Implementation Plan.

Note:

New generation tasks for file object/generation function combinations that are already contained in a plan can be added with the command IN of the function Modify implementation plan.

Select Implementation Plan from List - Code S

Implementation plans can be selected for further processing. Enter function code S in the File Implementation Menu and

- leave field Plan ID blank to display all plans for selection or
- enter a plan ID ending with an asterisk to specify a range of values.

The Implementation Plan Selection screen will appear:

```

13:48:24          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Implementation Plan Selection -

M  Impl. plan ID   Last modification   Exec. Date           Remarks
-  CHD             2002-07-30 at 13:59 2002-07-12 at 13:55  mod. after exec.
-  CHD-A-FDT       2002-07-03 at 13:48 2002-07-03 at 13:50
-  CHD-ABC         2002-07-03 at 13:25 2002-07-03 at 13:25  mod. after exec.
-  CHD-COPY2       2002-07-12 at 13:38 2002-07-12 at 13:41
-  CHD-FDT         2002-07-12 at 13:57 2002-07-03 at 13:14
-  CHD-FDT2        2002-07-03 at 13:15
-  CHD-NET         2002-07-07 at 18:09
-  CHD-TRUNC       2002-07-03 at 13:03 2002-07-03 at 13:03  mod. after exec.
-  CHDA            2002-07-02 at 13:04 2002-07-02 at 13:05
-  CHDABC          2002-07-02 at 13:22 2002-07-02 at 13:22
-  CHDAN           2002-07-08 at 13:47 2002-07-08 at 13:47
-  CHDB            2002-07-02 at 19:01
-  CHDBA           2002-07-06 at 09:54 2002-07-06 at 09:55
-  CHDBBB          2002-07-12 at 18:27
-  CHDC            2002-07-05 at 13:12          currently modified
-  CHDCOB          2002-07-12 at 13:03 2002-07-12 at 13:03

-  ( '.' = EXIT , 'T' = TOP )
    
```

The following information is displayed:

- implementation plan ID
- date and time of last modification
- execution date and time
- remarks (see below)

Enter a non-blank character in column M to select a plan for further processing.

Remarks	
The following remarks are possible:	
currently modified	Another user is currently modifying this implementation plan or the last attempt to modify the plan terminated abnormally.
mod. after exec.	The implementation plan has been modified since it was last executed.
executing	The plan is currently executing or the execution of the implementation plan terminated abnormally.

Execute Implementation Plan - Code X

The function Execute Implementation Plan can also be executed in batch mode. See Executing and Displaying an Implementation Plan in Batch Mode.

Note:

Plans containing a generation task for an Adabas file, ADAINV cards, a DDM or Vista translation tables can only be executed if the current virtual machine/network combination and the virtual machine/network

combination stored with the plan are identical.

```

09:57:05          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan 0            - File Implementation Menu -                    Profile HNO

Function                                Function

A Add an implementation plan            D Display implementation plan
C Copy implementation plan              E Extend implementation plan
M Modify implementation plan            S Select implementation plan from list
N Rename implementation plan            X Execute implementation plan
P Purge implementation plan             R Reexecute implementation plan

+-----+
!           File Implementation performed for           !
F !
! File ID ... CHD-COND                               !
I ! Language .. Language C                           !
C ! Library ... CCCLIB      Member ...                !
! DBnr .....          File nr ...                     !
!
C +-----+
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJl Flip Print Impl AdmFi SelFi Prof Main

```

When an Implementation Plan is executed, all generation tasks of status no or reexecute are executed. All other generation tasks are skipped. The task currently being executed is displayed in a window (see screen above). If the plan does not contain any executable task (status no or reexecute), a corresponding message is given.

You can repeat the execution of a plan if necessary (for example due to abnormal termination or system errors).

Reexecute Implementation Plan - Code R

This function consists of two parts:

- **Tasks are set to status reexecute**

All tasks in the implementation plan for which the command RE could be entered when executing the function Modify Plan are set to reexecute. The command RE is valid for all tasks that have been started at least once, i.e. in one of the following statuses: abended, error, executing, interrupted, performed, sav. error, or warning.

If a task is set to reexecute, any messages that were issued during the last generation are removed from the plan.

Generate DDM tasks are set to impossible if Natural Security is installed and a countersignature is required.

- **Function Execute plan is started**

In the second step, the function Execute Plan is started. If the execution terminates for any reason, the function Execute Plan can be started again at a later time to process the tasks that could not be executed.

Reexecute Implementation Plan in Batch Mode

This function can also be executed in batch mode. See Executing and Displaying an Implementation Plan in Batch Mode.

Error Handling - Online

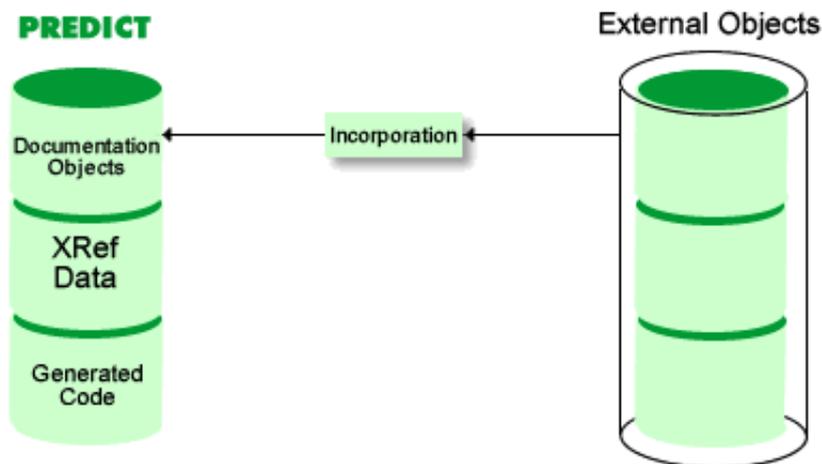
If parameters of a generation task cannot be applied, a message is returned and in some cases a selection window. Select a valid value from this window to execute the task.

A generation task that cannot be executed due to errors is given the status error. The status can be set to reexecute with the function RE or by modifying the generation options.

If a generation task terminates abnormally (status abended or executing), call the generation function directly to get a detailed error message (DICnnnn).

Incorporation

Predict objects can be created or updated by incorporating information from external objects, for example Natural DDMs or File Description Tables in Adabas .



This section covers the following topics:

- Concepts of Incorporation
- Calling Incorporation Functions
- Incorporating Adabas Databases
- Incorporating Adabas Files
- Incorporating DB2 Storagegroups
- Incorporating DB2 Databases
- Incorporating DB2 Tablespaces and SQL/DS DBspaces
- Incorporating DB2 and SQL/DS Tables/Views
- Incorporating NDBs for IMS Databases/Segments
- Incorporating Adabas Tables and Views
- Incorporating Natural DDMs
- Incorporating COBOL Copy Code
- Incorporating Super Natural Users
- Incorporating Natural Security Users
- Incorporating Tables / Views of SQL Database Systems

Concepts of Incorporation

Incorporation functions are used

- to find external objects that are not yet documented/connected to a documentation object,
- to document applications that are not yet documented in Predict,
- to redocument applications that are not completely documented in Predict.

Overview of Incorporation Options

Incorporation functions perform the following tasks:

- **Compare** external and documentation objects (command code T).
The function finds implemented objects already documented with Predict objects, where the implemented object and the Predict object are not connected.
If an implemented and a documented object with no differences are found and the parameter Connect automatic is set to Y, the implemented object is connected to the Predict object.
- **Connect** external and documentation objects (command code C).
The function connects documentation and implementation objects and - if appropriate - the documentation object is partially updated. For example: physical file attributes are updated, but file lists are not.
- **Display** IDs of objects that must be documented in Predict before objects can be incorporated (command codes M and V).
 - Display related DDM(s) that must be documented in Predict before DDMs can be incorporated (command code V)
 - Display DB2 databases, tablespaces and storagegroups that must be documented in Predict before tables/views or tablespaces can be incorporated (command code M).
- **Incorporate** external objects (command code I).
A Predict object is created, its attributes are taken from the implemented object. The implemented object and the new documentation object are connected.
- **Replace** documentation objects with newly created objects (command code R)
 - Replace IMS database objects in Predict with newly created objects.
 - Replace file object by incorporating COBOL copy code.

Overview of Incorporation Subfunctions

Not all incorporation subfunctions are available for all object types. See table below.

Documentation Object	Compare	Connect	Display Masters	Display Related DDMs	Incorporate	Replace
Adabas Database	Y	Y			Y	
Adabas File	Y	Y			Y	
DB2 Database	Y	Y			Y	
DB2 and SQL/DS Tables/Views	Y	Y	Y		Y	
DB2 Storagegroup	Y	Y			Y	
DB2 Tablespace	Y	Y	Y		Y	
DDM	Y	Y		Y	Y	
IMS Database					Y	Y
Natural Security User					Y	
SQL Tables/Views			Y		Y	
Super Natural User					Y	

Connecting External and Predict Objects

The incorporate functions Compare, Connect, Incorporate and Replace can be used to connect Predict objects and external objects.

See the section Handling of External and documentation Objects in this documentation for a description of the basic principles of connecting external and Predict objects.

Identifying Corresponding External and documentation Objects

Incorporation functions read information from Predict and from external sources to try and find objects that correspond. Corresponding objects are identified as follows:

Predict Object	Corresponding Object Identified with
Adabas Database	Physical database number (P-DBnr)
Adabas File	Physical database and file number (P-DBnr/P-Fnr) see also note below
DDM	DDM name - Predict file ID
DB2 Database	DB2 database name
DB2,SQL/DS Tables/Views	DB2,SQL/DS table/view name - Predict file ID
DB2 Storagegroup	DB2 storagegroup name
DB2 Tablespace	DB2 tablespace name
IMS Database	IMS database name
Natural SEC. User	User name - ID of User object in Predict
SQL Tables/Views	SQL table view name - Predict file ID
Super Natural User	User name - Predict user ID

Note:

With partitioned files, a Predict file object can be implemented with several physical files: In this case the database/file number of implemented files and corresponding Predict file objects need not be identical. Predict evaluates the Vista translation table to determine implemented files documented with Predict file object when the database/file numbers are not identical.

Sources of Information on External Objects

Predict reads the following external sources of information:

documentation Object	External Source of Information
Adabas Database	Adabas GCB
Adabas file	Adabas FCB and FDT
DB2 Database	DB2 catalog
DB2,SQL/DS Tables/Views	DB2 catalog
DB2 Storagegroup	DB2 catalog
DB2 Tablespace	DB2 catalog
DDM	DDM (in Predict system file)
IMS Database	Natural for DL1 database description (NDB) in Predict system file
Natural SEC. User	Natural Sec. system file
SQL Tables/Views	Catalog of corresponding DBMS
Super Natural User	Super Natural system file

Calling Incorporation Functions

When working online, incorporation functions are called from object type-dependent Incorporate screens. These are called by entering code I and an external object code in a Predict main menu.

To display a list of the valid external objects that can be processed with incorporation functions, select function code I in a Predict main menu and either leave the Object type blank or enter an asterisk. The external object types displayed depend on the environment. If DB2 is not installed, for example, DB2 objects are not contained in the list.

Incorporation Functions are Executed in Two Steps

- Create a list of external objects not yet connected to any Predict documentation object by specifying selection criteria.
- Process the selected objects with incorporation functions.

When incorporating Natural Security users and Super Natural users, the external object is selected and the incorporation subfunction is executed in one step.

Step 1: Selecting the Objects to be Processed

A selection list containing external objects is displayed. Selection criteria determine which objects are contained in the list. The layout of the list depends on the external object type.

```

13:50:11          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Incorporate DB2 Tablespaces -
Search-crit.: Tablespace name=ALL
  DB2      Tablespace   Predict
Cmd database name      dataspace ID      Remark
___ ABAG      ABAGTS01          DA missing
___ ABAG      ABAGTS02          DA missing
___ ABAG      ABAGTS03          DA missing
___ ABAG      ABAGTS04          DA missing
___ ABAG      ABAGTS05          DA missing
___ BGCGDA    AAB$PACE          SC missing
___ BGCGDA    TABSPACE      PD-TABSPACE    SC missing
___ DB01      TS01            DA missing
___ DSNCV     DSNAUTH        DA missing
    
```

Remarks which apply to all Object Types

Note:

Type-specific remarks are described with the respective external object types.

Remark	Function	Description
Equal	Compare	No significant differences between external and documentation object. Connection between the two can be established.
Different		Significant differences between external and documentation object. No connection between the two possible.
Connected	After execution of incorporation function Connect or Compare with parameter Connect automatic set to Y	Connection between external and documentation object was established.
Not connected		No connection established due to significant differences between external and documentation object.
Incorporated	After execution of function Incorporate	New Predict object created with information from external object.
Not incorporated		No new Predict object created.

Step 2: Processing Objects with Incorporation Functions

Compare - Code T

Compares the external objects with Predict objects and reports differences between attributes and lists of children (if applicable). Requires that both an implemented object and a corresponding Predict object exist. This function connects Predict objects to external objects if the following prerequisites are met:

- The parameter Connect automatic is set to Y.
- No differences or insignificant differences are found between the external object and the Predict object. If insignificant differences are found, the Predict object is updated.

Applicable to all external object types except Natural Security, Super Natural users and SQL tables/views.

Connect - Code C

Connects the external object to an existing Predict object and

- copies current information from external object to the Predict object;
- writes to the generation log of the object that a corresponding implemented database existed when the connection was established;
- connects child objects to the corresponding external objects.

Applicable to all external object types except Natural Security, Super Natural users and SQL tables/views.

Display Masters - Code M

When incorporating tables/views/tablespaces, higher-level objects (databases, storagegroups, tablespaces) must be documented in Predict.

This function finds higher-level objects that are not documented.

Display Related DDMs - Code V

When incorporating DDMs, related DDMs must already be documented in Predict. This function displays related DDMs that must be incorporated before a DDM can be incorporated.

Applicable to DDMs.

Incorporate - Code I

Creates Predict objects from information of external objects and connects the Predict objects to the external objects. Applicable to all external object types.

- When incorporating files/tables/views, Predict objects for the files in the database and links from files to fields are also created.
- When incorporating IMS databases, Predict objects for the files in the database and links from files to fields are also created.

Type-specific parameters determine how the different incorporation functions work.

Replace - Code R

This is the only function with which existing Predict objects documenting NDBs can be changed. Applicable to NDBs for IMS databases and segments.

Overview of Function Codes and Commands

The following table gives an overview of the codes to be entered in a Predict main menu and the corresponding commands.

Object Type	Code	Command
Adabas Database	AD	INCORPORATE ADABAS-DATABASE
Adabas File	AF	INCORPORATE FDT
Adabas D Table/View	BF	ESD-TABLE
COBOL Copy Code	CO	INCORPORATE COBOL
DDM	DD	INCORPORATE DDM
IMS Database	ND	INCORPORATE NDB
DB2 Database	D2	INCORPORATE DB2-DATABASE
Adabas table/view	EQ	INCORPORATE ESQ
Ingres Table/View	JF	INGRES-TABLE
Natural Security User	NS	INCORPORATE SECURITY
Oracle Table/View	OF	ORACLE-TABLE
Super Natural User	SU	INCORPORATE SUPER
DB2 Table/Views	T2	INCORPORATE TABLE
DB2 Storage space	SG	INCORPORATE STORAGEGROUP
DB2 Tablespace	TS	INCORPORATE TABLESPACE
Informix Table/View	XF	INFORMIX-TABLE
Sybase Table/View	YF	SYBASE-TABLE

Using Incorporation Functions in Batch Mode

Incorporation functions can be used both online and in batch mode.

Exception: Incorporate DDM is only available online.

Incorporating Adabas Databases

Prerequisites

- Adabas Online Services must be active.
- Databases must be Adabas Version 6 databases or above.
- If Natural Security is installed and function protection for Predict is activated, access rights must be given.

Restrictions

- Only databases accessible from the current virtual machine can be processed (see also the description of the Current VM parameter below).

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting Adabas Databases

The Incorporate Adabas Databases screen is called with function code I and object code AD in a Predict main menu or with the command INCORPORATE ADABAS-DATABASE.

```

09:49:24          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Incorporate Adabas Databases -

Current VM ..... HOME

Database number from ..... 1          (1-65535)
                  to ..... 65535     (1-65535)

Processing option
Connect automatic ..... N          (Y/N)
    
```

Parameters	
Current VM	Virtual machine of the database. A read-only field. The current virtual machine can be changed in the session profile. If no Current VM is specified in the session profile, the parameter Default current virtual machine in the Miscellaneous section of the General Defaults is used.
Database number from/to	The physical numbers of the Adabas databases to be incorporated. A range of up to 23 databases can be specified.
Connect automatic	Y The Adabas database and a corresponding Predict database object will be connected by the Compare function (code T) if no differences are detected between the file lists of the two objects.

Databases which meet the given selection criteria and are not already connected to a Predict database are then listed as follows:

```

13:59:54          ***** P R E D I C T 4.3.1 *****          2003-05-31
- Incorporate Adabas Databases -
Search-crit.: DBnr:8-30

Cmd  DBnr  Adabas name          Predict database ID          Remark
---  ---  ---
   8          HEB-ADA          Vista element exists
   9 GENERAL-DATABASE HEB-ADA-1
  10 DB010-NATURAL    BER-DA-10
  11          PD-A0000          Vista element exists
  12 GSD-MVS-012      DA with diff. type
  13          HEB-A-HEB-VM      DB error:3148
    
```

Columns in the List	
Cmd	Enter a valid incorporation subfunction. Enter an asterisk to display possible values.
DBnr	Number of the database.
Adabas name	The Adabas name of the database (the name specified with the ADADEF or the ADAREF utility).
Predict database ID	The IDs of Predict database object.
Remark	
Any of the following type-specific remarks or blank. For type-independent remarks see Remarks which apply to all Object Types.	
Note: If any of the following remarks is given, no incorporation function is possible.	
DB error: nnnn	Where nnnn is an Adabas response code between 3001 and 3255. The database is not defined in NTDB as type Adabas.
Vista element exists	Incorporating the database would destroy the consistency of the Predict documentation.
Database with different type exists	A Predict database object with a different database type exists. Incorporating the database would destroy the consistency of the Predict documentation.
Exists in other Network	Unique database number is required and the database already exists in another network. Incorporating the database would destroy the consistency of the Predict documentation.
Access via Vista	The database to be incorporated is only accessible using Vista, but Vista support is not activated in Predict.

Incorporation Functions for Adabas Databases

Compare - Code T

Compares (test) the database and the Predict database object and report differences between attributes and file lists.

If no differences between the file lists are found and Connect automatic is set to Y, connect the dictionary object to the database.

Connect - Code C

Connects the database to the existing Predict database object, and

- copies current information from Adabas to Predict object
- writes to the generation log of the object that a corresponding implemented database existed when the connection was established
- connects the checkpoint and security file (if any) of the Adabas database to Predict file objects in the file list of the database with the same file number.

Incorporate - Code I

Scope of the Incorporation

- A Predict database object of type A is created and connected to the external object.
- Vista elements for the database are created
 - if Vista is installed and
 - the Vista translation table contains translation elements with this database specified as target.
- The new database object is linked to the current virtual machine.
- If this database has a checkpoint or security file, the following files are added to the file list of the database: SAG-ADA-CHECKPOINT and SAG-ADA-Security.

Specify the following options in an additional input screen before executing the function.

Incorporation Options	
Database ID	The ID of the new database object in Predict (mandatory). The Adabas name of the database is displayed and can be accepted or overwritten. The Adabas name is specified in the ADADEF or the ADAREF utility when creating the database.
List incorporated code	Y The values of all attributes incorporated for the database are listed (optional).

Incorporate Adabas Database in Batch Mode

Command: INCORPORATE ADABAS-DATABASE

Field	Keyword	Position
Database number from	DBNR-FROM	1
Database number to	DBNR-TO	2
Connect automatic	AUTO-CONNECT	3
Function	SUB-FUNCTION	4
Database ID	DATABASE-ID	5
List incorporated code	LIST	6

Enter the parameters on the next line in positional or keyword form.

Example

To incorporate the Adabas database with number 188 as 'TEST-DB' and display the database attributes, enter:

```
INCORPORATE ADABAS-DATABASE  
DBNR-FROM=188,DBNR-TO=188,SUB-FUNCTION=I,DATABASE-ID=TEST-DB,LIST=Y
```

or

```
INCORPORATE ADABAS  
188,188,,I,TEST-DB,Y
```

The example above uses the Natural parameters ID=, and IA==

Incorporating Adabas Files

Prerequisites and Restrictions

- A Predict database object for the Adabas database of the file must exist.
- If Natural Security is installed and function protection for Predict is activated, access rights must be given.
- Certain features of the Connect and Incorporate functions require Adabas Online Services to be active and Adabas Version 6 or above (see descriptions below).

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

With partitioned or replicated files, a Predict file object can be implemented as several physical files. In this case the database/file number of implemented files and corresponding Predict file objects need not be identical. Predict evaluates the Vista translation table to determine if an implemented file is documented with a Predict file object, even if the database/file number are not identical.

Therefore, when incorporating partitioned files, only one implemented Adabas file can be incorporated using function I (incorporate). Other components of that file can then be connected with the incorporate function C (connect). Both the functions I and C update the Vista translation table of the file.

Selecting Adabas Files

The Incorporate Adabas Files screen is called with function code I and object code AF in a Predict main menu or with the command INCORPORATE FDT.

```

09:58:15          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Incorporate Adabas Files -

Current VM ..... HOME

Database number ..... 180      (1-65535)
Physical file number from ..... 1      (1-2096)
                        to ..... 255    (1-2096)

Select option
  Even if only FDT exists ..... Y      (Y/N)

Processing option
  Connect automatic ..... N      (Y/N)
  Incorporate file attributes ....* R      Required attr.
  Adabas password .....

```

Parameters	
Database number	The physical number of the Adabas database in which the Adabas files to be incorporated reside.
Physical file number	Range of physical file numbers of the Adabas files to be incorporated.
even if only FDT exists	<p>Y A file is to be incorporated even if it has been deleted with the Keep FDT option.</p> <p>N only valid if the following conditions are met:</p> <ul style="list-style-type: none"> ● Adabas Online Services must be available; ● The file to be incorporated must reside in a database of Adabas Version 6 or above; ● The file to be incorporated must not have been deleted with the Keep FDT option, since it must have implementation data.
Connect automatic	<p>Y The Adabas file and the Predict file object will be connected by the Compare function (code T) if no differences are detected between the field lists of the two objects.</p>
Incorporate file attributes	<p>Determines which attributes of the data dictionary file object are to be updated. Provided that Adabas Online Services are available and the file to be connected or incorporated resides in a database of Adabas Version 6 or above, the following values can be specified:</p> <p>R File attributes required when loading a file and any other file attributes specified in the data dictionary object</p> <p>O All file attributes including optional attributes not specified in the data dictionary object.</p>
Adabas password	The password required to access the file if the file is protected by Adabas security. The password is not converted to upper case.

Files which meet the given selection criteria and are not already connected to a Predict file object will then be listed as follows:

```

13:04:29          ***** P R E D I C T 4.3.1 *****                2003-05-31
                    - Incorporate Adabas Files -
Search-crit.: DBnr=180,File range=1-255,Even-only-FDT=Y

Cmd PFnr  Adabas name          Predict File ID          Remark
---
   5 MISCELLANEOUS
   6 N-213-FNATS             N-213-FNATS
   8 AUTOMOBILES
   9 FINANCE
  10 CST332-SYSF
    
```

Columns in the List	
Adabas name	The name that was specified, either to the ADALOD utility or to Adabas Online Services, when the file was loaded into the database.
Predict File ID	The ID of Predict file object of type A that may correspond to the Adabas files.
Remark	
Any of the following type-specific remarks or blank. In any of the following messages is displayed, the connect function cannot be executed. For type-independent remarks see Remarks which apply to all Object Types.	
Partitioned master missing	Documentation of Vista elements and entries in Vista translation table are different.
ISN range overlapping	Predict file object with physical distribution type expanded is different.
Different components	Predict file object with physical distribution type expanded is different. Vista elements and entries in Vista translation table are different.
Different Vista type	Documentation of Vista elements and entries in Vista translation are different.
Different local definition	Documentation of Vista elements and entries in Vista translation are different.

Incorporation Functions for Adabas Files

The functions T, C and I apply.

Compare - Code T

Compares Adabas files and corresponding Predict file objects.

Differences between field lists and - if Adabas Online Services are available - their file attributes are reported. If no differences between the field lists are found and Connect automatic is set to Y, the Adabas file and the Predict file object are connected.

Connect - Code C

Connects Adabas file to the corresponding Predict file object and

- copies information from the Adabas FCB to the Predict file object,
- writes to the generation log of the object that a corresponding Adabas file existed when the connection was established.
- adds or updates Vista elements according to Vista translation table (if applicable).

Incorporate - Code I

Scope of the Incorporation

- One Predict file object is created for each Adabas file.
The Predict file objects are connected to the Adabas files.
- The new file object is linked to the database object via association *Contains FI*.
- One field object is created for each field in the file.
The ID assigned to each field object is created from the name of the field followed by either -GROUP for a group or -FIELD for a single field.
A prefix and/or suffix specified with the parameters Field name prefix/suffix is/are added.

- If the file is physically coupled, a relationship object is created for each coupling definition.
- One Vista element (if applicable). If the file is accessible via several entries in the Vista translation table, the Vista element is created for the lowest logical file number.
- If Adabas Online Services are active and the file is implemented in a database of Adabas Version 6 or above, information on the implementation of the file (space allocation, for example) is incorporated.

Before the function is executed, enter the following parameters in an additional input screen. Parameter File ID is mandatory, all other parameters are optional.

Incorporation Options	
File ID	ID of the new file object (mandatory). The Adabas name of the file is displayed and can be accepted or overwritten.
Field name prefix	Prefix of each field ID.
Field name suffix	Suffix of each field ID.
List incorporated code	Y The values of all attributes incorporated for the file are to be listed.
Edit elementary fields	Y The function Edit elements of a file is to be executed after incorporation.

Incorporating Adabas Files in Batch Mode

Command: INCORPORATE FDT

Enter parameters on next line in positional or keyword form.

Field	Keyword	Position
Database number	DBNR	1
Physical file number from	FROM-FNR	2
Physical file number to	TO-FNR	3
Even if only FDT exists	EVEN-ONLY-FDT	4
Connect automatic	AUTO-CONNECT	5
Incorporate file attributes	FILE-ATTR	6
Adabas password	PSW	7
Function	SUB-FUNCTION	8
File ID	FILE-ID	9
Field name prefix	PREFIX	10
Field name suffix	SUFFIX	11
List incorporated code	LIST	12

Example

To incorporate the file 1 in database 177 and give the file the name 'INC-TEST', code the command:

Incorporating Adabas Files

Incorporating Adabas Files in Batch Mode

```
INCORPORATE FDT  
DBNR=177, FROM-FNR=1, TO-FNR=1, SUB-FUNCTION=I,  
FILE-ID=INC-TEST
```

or (abbreviated)

```
INCORPORATE FDT  
177,1,1,,,,,I, INC-TEST
```

The example above uses the Natural parameters ID=, and IA==

Incorporating DB2 Storagegroups

Prerequisites

Incorporation of DB2 objects is subject to DB2 security mechanisms. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation**.

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting DB2 Storagegroups

The Incorporate DB2 Storagegroup screen is called with function code I and object code SG in a Predict main menu or with the command INCORPORATE STORAGEGROUP.

```

09:31:44          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Incorporate DB2 Storagegroups -

Storagegroup name ....

Select option
  Creator name .....

Processing option
  Connect automatic .. N (Y,N)
    
```

Parameters	
Storagegroup name	DB2 storagegroup name. Asterisk notation is allowed.
Creator name	DB2 creator name. Asterisk notation is allowed.
Connect automatic	Y With the Compare function: the DB2 storagegroup and the Predict object are connected if no differences are detected between the two objects.

```

13:07:27          ***** P R E D I C T  4.3.1  *****          2003-05-31
          - Incorporate DB2 Storagegroups -
Search-crit.: Storagegroup name=ALL
Storage-      Predict
Cmd Creator  group name  storagespace ID          Remark
___ BHD      LMRK0001
___ BHD      RISPASTO
___ BHD      SYSDEFLT    SYSDEFLT
___ RSH      GROUPY
___ RSH      RSH#GROU
___ RSH      RSHGROU
___ SCM      SCMSTG
    
```

Columns in the List	
Creator	Creator of the storagegroup.
Storagegroup name	Name of the storagegroup.
Predict Storagegroup ID	ID of a Predict storagegroup documenting the storagegroup.
Remark	A type-independent remark (see Remarks which apply to all Object Types or blank).

Incorporation Functions for DB2 Storagegroups

The functions T, C and I apply.

Note:

Corresponding DB2 storagegroups and Predict objects of type storagegroup are identified by the DB2 storagegroup name.

Compare - Code T

Compares the storagegroup and the Predict storagegroup and reports differences between attributes.

If no differences between the two objects are found and Connect automatic is set to Y, the dictionary object is connected to the storagegroup.

Connect - Code C

Connects the storagegroup to the existing Predict storagegroup, and

- updates the Predict storagegroup;
- writes to the generation log of the Predict object that a corresponding implemented storagegroup existed when the connection was established.

Incorporate - Code I

Scope of the Incorporation

- Creates a Predict storagegroup with attributes documenting the storagegroup.
- Connects the storagegroup and the Predict storagegroup.

Before the function is executed, specify the following parameters in an additional input screen. Storagegroup ID is mandatory, List incorporated code is optional

Incorporation Options	
Storagegroup ID	ID to be assigned to the new Predict storagegroup.
List incorporated code	Y The values of all attributes incorporated for the table or view are to be listed.

Incorporate DB2 Storagegroups in Batch Mode

Command: INCORPORATE STORAGEGROUP

Enter parameters on next line using positional or keyword form.

Field	Keyword	Position
Storagegroup name	STOGROUP	1
Creator name	CREATOR	2
Connect automatic	AUTO-CONNECT	3
Function	SUB-FUNCTION	4
Storagespace ID	STOSPACE-ID	5
List incorporated code	LIST	6

Example

To incorporate the DB2 storagegroup 'SYSDEFLT' with storagespace ID 'TEST-ST' and display the storage space attributes, code the command:

```
INCORPORATE STORAGEGROUP
STOGROUP=SYSDEFLT , SUB-FUNCTION=I , STOSPACE-ID=TEST-ST , LIST=Y
```

or

```
INCORPORATE STORAGEGROUP
SYSDEFLT , , I , TEST-ST , Y
```

The example above uses the Natural parameters ID=, and IA==

Incorporating DB2 Databases

Prerequisites

Incorporation of DB2 objects is subject to DB2 security mechanisms. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation**.

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting DB2 Databases

The Incorporate DB2 database screen is called with function code I and object code D2 in a Predict main menu or with the command INCORPORATE DB2.

```

09:23:24          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Incorporate DB2 Databases -

DB2 database name ....

Select option
  Creator name .....

Processing option
  Connect automatic .. N (Y/N)
    
```

Parameters	
DB2 database name	DB2 database name. Asterisk notation is allowed.
Creator name	Creator name. Asterisk notation is allowed.
Connect automatic	Y The DB2 database and the Predict object will be connected by the Compare function (code T) if no differences were detected between the file lists of the two objects.

Databases which meet the selection criteria and are not already connected to a Predict database object will then be listed as follows.

```

13:09:09          ***** P R E D I C T 4.3.1 *****          2003-05-31
- Incorporate DB2 Databases -
Search-crit.: DB2 database name=ALL
Database Default
Cmd Creator  name      stogroup Predict database ID      Remark
___ BHD      ABAG      SYSDEFLT
___ BHD      ADAPT1   SYSDEFLT
___ BHD      DSNCV    SYSDEFLT
___ BHD      DSNDB07  SYSDEFLT
___ BHD      DSNDDF   SYSDEFLT
___ BHD      DSNRGFDB SYSDEFLT
___ BHD      DSNRLST  SYSDEFLT
___ BHD      LANDMARK LMRK0001      SC missing
___ BHD      RISPADB  RISPASTO      SC missing
    
```

Columns in the List	
Creator	Creator of the database.
Database name	Name of the database
Predict database ID	ID of Predict database object for a DB2 database with this DB2 name.
Remark	
Either the following type-specific remark or blank. For type-independent remarks see Remarks which apply to all Object Types.	
SC missing	Default storagegroup is not documented. Incorporation is not possible.

Incorporation Functions for DB2 Databases

The functions T, C and I apply.

Compare - Code T

Compares the database and the Predict database object and reports differences between attributes and file lists.

If no differences between the file lists are found and Connect automatic is set to Y, the dictionary object is connected to the database.

Connect - Code C

Connects the DB2 database to the corresponding Predict database object and

- copies current information from the DB2 catalog into the dictionary object;
- writes to the generation log of the Predict database object that a corresponding implemented database existed when the connection was established.

Incorporate - Code I

Scope of the Incorporation

- Creates a database object of type D with attributes documenting the database.
- The new database object is linked to the Predict object documenting the current virtual machine via association *Contains DA*.

- Connects the database and the Predict database object.

Before the function is executed, specify the following parameters in an additional input screen. The parameter Database ID is mandatory, the other parameter optional

Incorporation Options	
Database ID	ID of the new Predict database object. The creator and DB2 name of the database joined by a hyphen is displayed and can be accepted or overwritten.
List incorporated code	Y the values of all attributes incorporated for the database are to be listed.

Incorporate DB2 Databases in Batch Mode

Command: INCORPORATE DB2-DATABASE

Enter parameters on next line using positional or keyword form.

Field	Keyword	Position
DB2 database name	DB2NAME	1
Creator name	CREATOR	2
Connect automatic	AUTO-CONNECT	3
Function	SUB-FUNCTION	4
Database ID	DATABASE-ID	5
List incorporated code	LIST	6

Example

To incorporate the DB2 database 'DSN8D13A' as database object in Predict with ID 'TEST-DB' and display the database attributes, code the command:

```
INCORPORATE DB2
DB2NAME=DSN8D13A , SUB-FUNCTION=I , DATABASE-ID=TEST-DB , LIST=Y
```

or

```
INCORPORATE DB2
DSN8D13A , , I , TEST-DB , Y
```

The example above uses the Natural parameters ID=, and IA==

Incorporating DB2 Tablespaces and SQL/DS DBspaces

Prerequisites

Incorporation of DB2 objects is subject to DB2 security mechanisms. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation**.

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting Tablespaces

The Incorporate DB2 Tablespaces screen is called with function code I and object code TS in a Predict main menu or with the command INCORPORATE TABLESPACE.

```

09:28:15          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Incorporate DB2 Tablespaces -

Tablespace name .....

Select options
  Creator name .....
  DB2 database name ...

Processing option
  Connect automatic ... N (Y,N)
    
```

Parameters	
Tablespace name	Name of DB2 tablespace. Asterisk notation is allowed.
Creator name	Name of tablespace creator. Asterisk notation is allowed.
DB2 database name	Name of DB2 database. Protected when processing SQL/DS DBspaces. Asterisk notation is allowed.
Connect automatic	Y The DB2 Tablespace and the Predict object will be connected by the Compare function (code T) if no differences are detected between the file lists of the two objects.

```

13:13:06          ***** P R E D I C T 4.3.1 *****          2003-05-31
          - Incorporate DB2 Tablespaces -
Search-crit.: Tablespace name=ALL
  DB2      Tablespace   Predict
Cmd database name      dataspace ID      Remark
___ ABAG      ABAGTS01          DA missing
___ ABAG      ABAGTS02          DA missing
___ ABAG      ABAGTS03          DA missing
___ ABAG      ABAGTS04          DA missing
___ ABAG      ABAGTS05          DA missing
___ BGCD A    AAB$PACE          SC missing
___ BGCD A    TABSPACE      PD-TABSPACE      SC missing
___ DB01      TS01          DA missing
    
```

Columns in the List	
DB2 database	Name of the database containing the tablespace.
Tablespace name	Name of the tablespace.
Predict dataspace ID	ID of the Predict dataspace object documenting the DB2 tablespace.
Remark	
Any of the following type-specific remarks or blank. For type-independent remarks see Remarks which apply to all Object Types.	
DA missing, ST missing	The DB2 database or storagegroup are not documented in Predict. Incorporation is not possible.

Incorporation Functions for Tablespaces

The functions T, C, M and I apply.

Compare - Code T

Compares DB2 tablespaces and corresponding Predict dataspace objects. Differences between file lists and tablespace/dataspaces attributes are reported.

If no differences are found and Connect automatic is set to Y, the tablespace/DBspace and the Predict dataspace object are connected.

Connect - Code C

Connects DB2 tablespaces to Predict dataspace object and

- copies current information from the DB2/SQL/DS catalog to the dictionary object
- writes to the generation log of the Predict dataspace object that a corresponding implemented database existed when the connection was established.

Display Masters - Code M

The DB2 databases and storagegroups a tablespace is related to must be documented in Predict before the tablespace can be incorporated. The incorporation function M can be used to identify all objects that are missing. These can then be incorporated before the tablespace is incorporated.

Incorporate - Code I

Scope of the Incorporation

- A Predict dataspace object is created for each tablespace and connected to the tablespace.
- The dataspace object is linked to the corresponding database in Predict via association *Contains DC*.

Before the function is executed, enter the following parameters in an additional input screen. Dataspace ID is mandatory, List incorporated code is optional

Incorporation Options	
Dataspace ID	ID of the Predict dataspace object. The DB2 tablespace name is displayed and can be accepted or overwritten.
List incorporated code	Y Display what has been incorporated as attributes of the dataspace object.

Incorporate DB2 Tablespaces in Batch Mode

Command: INCORPORATE TABLESPACE

Enter parameters on next line using positional or keyword form.

Field	Keyword	Position
Tablespace name	TBSPACE	1
Creator name	CREATOR	2
DB2 database name	DB2NAME	3
Connect automatic	AUTO-CONNECT	4
Function	SUB-FUNCTION	5
Dataspace ID	DSSPACE-ID	6
List incorporated code	LIST	7

Example

To incorporate the DB2 Tablespace with the DB2 tablespace name 'SYSDBASE' with dataspace ID 'TEST-TS' and display the dataspace attributes, code the command:

```
INCORPORATE TABLESPACE
TBSPACE=SYSDBASE , SUB-FUNCTION=I , DSSPACE-ID=TEST-TS ,
LIST=Y
```

or

```
INCORPORATE TABLESPACE
SYSDBASE , , , I , TEST-TS , Y
```

Note: The example above uses the Natural parameters ID=, and IA==

Incorporating DB2 and SQL/DS Tables/Views

Prerequisites

- For DB2 tables and views, a Predict object documenting the DB2 database containing the tables/views must exist.
- For SQL/DS tables and views, a corresponding database object in Predict is not required. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation**.

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting DB2 and SQL/DS Tables/Views

The Incorporate DB2 Tables/Views screen is called with function code I and object code T2 in a Predict main menu or with the command INCORPORATE TABLES.

09:30:17	***** P R E D I C T 4.3.1 *****	2003-05-31
Plan 0	- Incorporate DB2 Tables/Views -	
Table / View name		
Select options		
Tables or Views	*	
Creator name		
DB2 database name ...		
Processing option		
Connect automatic ...	N (Y,N)	

Parameters	
Table / view name	The name in DB2 of the tables and views from which the data dictionary objects are to be created. Asterisk notation is allowed.
Tables or views	Limit the function to files of one type. Valid values: T DB2 tables V DB2 views
Creator name	The DB2 "creator" of the tables and views from which the data dictionary objects are to be created. Asterisk notation is allowed.
DB2 database name	The name in DB2 of the database or databases in which the specified tables and views reside. Asterisk notation is allowed. When incorporating Tables/Views from SQL/DS, this parameter is protected.
Connect automatic	Y DB2 table/views and the Predict object will be connected by the Compare function (code T) if no differences were detected between the field lists of the two objects.

Files which meet the given selection criteria and are not already connected to a Predict file object are listed as below.

```

13:03:38          ***** P R E D I C T  4.3.1  *****                2003-05-31
                    - Incorporate DB2 Tables/Views -
Search-crit.: Table/View name=ALL,DB name=ALL,Creator=ALL
                    Doc.
Cmd Creator   Table/View name      Database  T   exists   Remark
___ ABA       DEMO1                DSND04   T   No
___ ABA       PLAN_TABLE             DSND04   T   No
___ AZ        APPT_TODAY             DSND04   V   No
___ AZ        D_BAR                  PREDICT   T   No
___ AZ        D_FX                   PREDICT   T   No
___ AZ        D_NAT1                 DSND04   T   Yes
    
```

Columns in the List	
Creator	Creator in DB2.
Table/view name	Name of the table/view.
Database	Name of DB2 database(s) where the table/view is implemented.
T	Type of the DB2 file: T DB2 table V DB2 view
Doc. exists	Y Predict object exists for the table/view.
Remark	
Any of the following type-specific remarks or blank. For type-independent remarks see Remarks which apply to all Object Types.	
DA missing, ST missing, DS missing	The DB2 database, storagegroup or tablespace the table/view belongs to are not documented in Predict. Incorporation is not possible.
Master missing	The specified DB2 view cannot be incorporated as a data dictionary object until a dictionary object for a related DB2 table or view has been created.
Invalid file-type	The type of the specified DB2 table or view does not match the type of the corresponding Predict object: the file type of the data dictionary object is either not D (DB2 table) or not E (DB2 view).

The tables/views in the list can then be processed by entering a function code in the Cmd column. The functions T, C and I apply.

Incorporation Functions for Tables/Views

The functions T, C, M and I apply.

Note:

Corresponding DB2 tables/views and Predict file objects of type D and E are identified by DB2 table/view name.

Compare - Code T

Compares DB2 table/views and corresponding Predict file object. Differences between field lists and attributes of the tables/views are reported.

If no differences between the field lists are found and Connect automatic is set to Y, the DB2 tables/view and the Predict file object are connected.

Connect - Code C

Connects DB2 tables/views to Predict file object and

- copies current information from the DB2 catalog to the dictionary object;
- writes to the generation log of the Predict file object that a corresponding implemented table/view existed when the connection was established.

Display Masters - Code M

DB2 databases, storagegroups and tablespace and tables/views a table/view is related to must be documented in Predict before the table/view can be incorporated.

This function finds all objects that are not documented. These can then be incorporated before the table/view is incorporated.

Incorporate - Code I

Scope of the Incorporation

- One Predict file object is created for each table or view.
The ID of the new file object consists of the creator name and DB2 name of the table or view joined by a hyphen.
- One field object is created for each field in the table or view.
The ID assigned to each new field object will be the name of the field in DB2. For each view that is incorporated, the appropriate parts of the subselect specification in the SQL statement CREATE will be copied to the file and field objects created.
- One field object is added to file SAG-DISTINCT-TYPE for each distinct type that is not already documented.
The ID assigned to each new field object consists of the schema name and the type name concatenated by a hyphen.
The connection between the standard fields of SAG-DISTINCT-TYPE and the table field is established if the field in DB2 is distinct type.
- One Predict file object with file type IV (Intermediate view) is created for each subselect clause in the from clause of a view definition. For each column in the select list of subselect a Predict field object is created. For each intermediate view that is created, the appropriate parts of the subselect specification in the SQL statement CREATE will be copied to the file and field objects created.
- One Predict field object (type SP) for each composite index is created.
- One Predict field object (type SP) for each composite foreign key is created if there is not already a Predict field object for a composite index built from the same columns.
- One Predict relationship object is created for each foreign key.
- Labels defined for columns are documented as field headers.
- A trigger object is added for each trigger defined for a table.
A link is established either to the file object via association *Has TR* or to the corresponding field objects in case of an update trigger.
- A verification is added and linked to the corresponding field for each check constraint that is defined for one column.
- Attributes of the corresponding file object are updated for each table check constraint (check expression for more than one column).

Calling the Function

Before the function is executed, enter the following parameters in an additional input screen:

Incorporation Options	
Comments as abstract	<p>Y String constants assigned with COMMENT ON are to be incorporated as the abstract of the corresponding Predict object:</p> <ul style="list-style-type: none"> ● Comments assigned to the table or view are to be copied to the file object. ● Comments assigned to a field of the table or view are to be copied to the field object.
List incorporated code	<p>Y The values of all attributes incorporated for the table or view are to be listed.</p>

Incorporating DB2 Tables and Views in Batch Mode

Command: INCORPORATE TABLE

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
Table/View name	TABLE	1
Tables or Views	TYPE	2
Creator name	CREATOR	3
DB2 Database name	DB2NAME	4
Connect automatic	AUTO-CONNECT	5
Function	SUB-FUNCTION	6
Incorporate comments	COMMENT	7
List incorporated code	LIST	8

Note:

The keyword ABSTRACT can be used as a synonym for COMMENT.

Example

To incorporate the table 'TDEPT' in database 'DSN8130', code the command:

```
INCORPORATE TABLE
TABLE=TDEPT,TYPE=T,DB2NAME=DSN8130,,SUB-FUNCTION=I
```

or

```
INCORPORATE TABLE
TDEPT,T,DSN8130,,I
```

The example above uses the Natural parameters ID=, and IA==

Incorporating NDBs for IMS Databases/Segments

NDBs and UDFs of physical and logical IMS databases can be incorporated into the Predict data dictionary.

As a safeguard against accidental overwriting, an NDB will not be incorporated if the dictionary already contains a database incorporated from the same NDB. To change a database object of type I, use the Replace function (code R).

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting Natural DBDs for IMS Databases and Segments

The Incorporate NDBs screen is displayed by selecting function code I and object code ND in a Predict main menu or by entering the direct command INCORPORATE NDB.

```

13:44:22          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Incorporate NDBs -

NDB name .....
```

Parameters	
NDB name	Name of the NDB. NDB names correspond to the name of IMS databases. Asterisk notation is allowed.

NDBs which meet this selection criterion are then listed as follows:

```

13:27:58          ***** P R E D I C T  4.3.1  *****          2003-05-31
          - Incorporate NDBs -

Search-crit.:NDB name=ALL
                Segment
Cmd NDB name NDB type Count Database ID          Remark
___ ARTICLE Physical      6  ARTICLE4
___ CUSTOMER Physical     3  CUSTOMER
___ ED00DBD Physical     5  ED00DBD1
___ PARTLIST Logical      3  PARTLIST2
___ PARTUSED Logical      3  PARTUSED
___
```

Columns in the List	
NDB name	Name of the NDB corresponding to an IMS database.
NDB type	Type of the IMS database: either Physical or Logical.
Segment count	The number of segments in the IMS database.
Database ID	The ID of a Predict database object documenting the NDB.
Remark	After successful execution of the function: either Replaced or Incorporated.

Incorporating Natural DBDs for IMS Databases and Segments

Incorporate - Code I

Scope of the Incorporation

- A database object of type I is created for each NDB.
- A Predict file object of type I is created for each segment of the NDB.
- Predict file objects of type J are created if a segment has UDFs.
If the segment contains a DBID and FNR (connection to the UDF) Predict checks whether a UDF is defined with this DBID and FNR and whether the DBID is defined in the NTDB macro as an IMS/DL1 database.
If no corresponding UDF is defined, an error message is returned.
- If the DBID does not correspond to an IMS/DL1 database, any UDF is incorporated with a warning message. The DBID and FNR remain unchanged.
- If a corresponding UDF is defined, one or more files of type J containing all user defined fields are created.

Each time a redefinition is found, i.e., the offset of a field is equal to or less than the offset of a previous field, a new file of type J is created. Gaps between two user defined fields are filled with DUMMY fields.

Predict ensures that the Adabas short names of UDFs in logical child segments fall within the range allowed by Natural for DL1.

This range is documented in the DDA Services and should only be changed on special advice of SAG. DBID/FNR of the UDF are recorded in the generation log of the file of type I.

Before the Incorporate function is executed, specify the following parameters for each NDB in an additional input screen:

Incorporation Options	
Segment name prefix	The prefix to be concatenated both with the NDB name to create an ID for the Predict database object and with the name of each segment in the NDB to create an ID for the corresponding Predict file object of type I.
Segment name suffix	The suffix to be concatenated both with the NDB name to create an ID for the Predict database object and with the name of each segment in the NDB to create an ID for the corresponding Predict file object of type I.
Layout name prefix	The prefix to be concatenated with the name of each segment in the NDB that has user defined fields, to create an ID for the corresponding Predict file object of type J.
Layout name suffix	The suffix to be concatenated with the name of each segment in the NDB that has user defined fields, to create an ID for the corresponding Predict file object of type J. If several layouts are created for one segment, a different number is added to the layout name suffix for each layout.
Field name prefix / suffix	How each segment name is to be concatenated with the name of each field in that segment (file of type I): P as a prefix to the field name S as a suffix to the field name.
Only for identical field names	Y Prefixes and suffixes are only used to distinguish between fields with identical names in the same NDB.
List incorporated code	Y the resulting definitions are to be displayed.
Standard comments	Y The first three abstract lines of each object incorporated (database, files and fields) will contain the name of the incorporated NDB and its date of incorporation. N The incorporated Predict objects will contain no abstract.

Replace - Code R

Replaces Predict objects documenting NDBs.

Prerequisites and Restrictions

Predict objects documenting NDBs will be not be replaced in the following circumstances:

- if a physical database is to replace a logical database or vice versa;
- if a file of type I or a related file of type J or K for which a DDM exists is to be deleted and Natural Security does not allow the user to delete the DDM.

Scope of the Replace Operation

The Replace function creates, replaces and deletes Predict file objects.
The following rules apply:

- New Predict file objects will be **created** for all segments in the NDB that are not yet documented in Predict.
- Existing Predict file objects are **replaced** if they document segments that are still used in the NDB. Changes to IMS fields and deletion of IMS fields are then rippled across all related files of types J and K that contain these fields.

The following files will be marked as modified after generation after this rippling:

- If the changed file represents a segment of a physical database:
 - the changed file and all its related files of types J and K;
 - Files of type I that depend on the changed file and all their related files of types J and K.
- If the changed file represents a segment of a logical database:
 - the changed file and all files of types J and K that are related to the physical file of type I from which the changed file is derived;
 - Files of type I that depend on the changed file and all files of types J and K that are related to the physical files of type I from which these files of type I are derived.
- Predict file objects documenting segments that are no longer contained in the NDB will be **deleted**. Whenever a file of type I is deleted, all related files of types J and K are also deleted. If DDMs have been generated from the files of type I, J and K, these files can only be purged if the user is authorized to modify the Natural Security definition of the files. This is only applicable if Natural Security is active. See Protecting DDMs in the section **Protecting External Objects in Predict with Natural Security** in the **Predict Security documentation**.
- Files of type J are not replaced, even if the UDF has been changed.

Before the Replace function is executed, enter the following parameters for each NDB in an additional input screen:

Replace Options	
Keep layout structure	<p>When files of type I are replaced, changes in IMS fields are rippled across the layouts and userviews that contain them. Rippled changes to the offset and length of IMS fields in a file of type J and deletion of IMS fields from a file of type J can change the offset of fields later in a layout.</p> <p>Y Predict either adjusts the length of DUMMY fields to cancel this effect or issues a warning that the effect cannot be cancelled.</p>
List incorporated code	<p>Y The resulting definitions are to be displayed.</p>
Comments as abstract	<p>Y The first three abstract lines of each object incorporated (database, files and fields) will contain the name of the incorporated NDB and its date of incorporation.</p> <p>N The incorporated Predict objects will contain no abstract.</p> <p>Before replacing any files of type I during an online session, Predict displays the file IDs together with the IDs of related files of types J and K.</p> <p>Confirm the deletions to continue processing.</p>

Incorporating IMS Database in Batch Mode

Command: INCORPORATE NDB

Enter parameters on the next line in positional or keyword form.

Field / Code	Keyword	Position
NDB name	NDB	1
I (incorp.) o. R (replace)	SUB-FUNCTION	2
Segment name prefix	SEGMENT-PREFIX	3
Segment name suffix	SEGMENT-SUFFIX	4
Layout name prefix	LAYOUT-PREFIX	5
Layout name suffix	LAYOUT-SUFFIX	6
Field name prefix / suffix	FIELD-NAME	7
Only identical field names	ID-ONLY	8
List incorporated code	LIST	9
Keep layout structure	KEEP	10
Comments as abstract	COMMENT	11
Delete	DELETE	12

Note:

The keyword ABSTRACT can be used as a synonym for COMMENT.

Example

To incorporate the IMS-NDB of name TEST, code the command:

```
INCORPORATE NDB
NDB=TEST, SUB-FUNCTION=I
```

or (using positional parameters)

```
INCORPORATE NDB
TEST, I
```

The example above uses the Natural parameters ID=, and IA==

Extra Parameter for Replacing an NDB in Batch Mode	
DELETE	<p>N</p> <p>If a file of type I is to be replaced, no changes are made to Predict; the IDs of the files of type I that should have been replaced are listed. The function can then be rerun with DELETE=Y.</p> <p>This parameter is only available in batch mode. Default is Y.</p>

Incorporating Adabas Tables and Views

Prerequisites

See the section Adabas SQL Server in the **Predict and Other Systems** documentation.

Note:

See also Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting Adabas Tables/Views

```

10:00:30          ***** P R E D I C T  4.3.1  *****                2003-05-31
Plan    0          - Incorporate Adabas Tables/Views -

Table / View name ....

Select options
Tables or Views ...*
Creator name .....

Processing option
Connect automatic .. N (Y,N)
    
```

Parameters	
Table / view name	The name in Adabas SQL Server of the tables and views from which the data dictionary objects are to be created. Asterisk notation is allowed.
Tables or views	Limit the function to files of one type. Valid values: T Adabas tables (files of type A(SQL) or Adabas SQL Server base tables (files of type AT or A(SQL)). V Adabas views
Creator name	The schema name of the tables and views in Adabas SQL Server from which the data dictionary objects are to be created. Asterisk notation is allowed.
Connect automatic	Y Adabas table/views and the Predict object are connected by the Compare function (code T) if no differences were detected between the field lists of the two objects.

```

13:09:11          ***** P R E D I C T  4.3.1  *****          2003-05-31
                    - Incorporate Adabas Tables/Views -
Search-crit.: Table/View name=ALL,Creator=ALL

                               Doc.
Cmd Creator          Table/View name  T exists Remark
--- ESQ             FILE150           T No
--- ESQ             FILE151           T No
--- ESQ             FILE152           T No
--- ESQ             FILE153           T No
    
```

Columns in the List	
Creator	Schema ID in Adabas SQL.
Table/view name	Name of the table/view.
T	Type: T Adabas table V Adabas view
Doc. exists	Y Predict object exists for the table/view.
Remark	
Any of the following type-specific remarks or blank. For type-independent remarks see Remarks which apply to all Object Types.	
Master missing	The specified Adabas view cannot be incorporated in Predict until an object for a related Adabas file or Adabas view has been created, or file of type AT cannot be incorporated because the related Adabas file is not documented in Predict.
Different file type	The type of the specified Adabas table or view does not match the type of the corresponding Predict object: the file type of the data dictionary object is either not A (Adabas file) or not B (Adabas view).
Lev0 table exists	The base table in Adabas SQL Server has level 0, a corresponding Adabas file and an AT file not at level 0 exist in Predict, but the files have different IDs.

Incorporation Functions for Tables/Views

Note:

Corresponding Adabas tables/views and Predict file objects of type A and B are identified by file ID.

Compare - Code T

Compares Adabas table/views and corresponding Predict file object.
Differences between field lists and attributes of the tables/views are reported.

If no differences between the field lists are found and Connect automatic is set to Y, the Adabas table/view and the Predict file object are connected.

Connect - Code C

Connects Adabas tables/views to Predict file object and copies current information from the Adabas SQL Server directory to the dictionary object.

Writes to the generation log of the Predict file object that a corresponding implemented table/view existed when the connection was established.

Display Masters - Code M

A higher level object must be documented in Predict before the Adabas view can be incorporated.

The function Display Masters finds all tables/views that are not documented but must be documented before table/views can be incorporated.

Incorporate - Code I

Scope of the Incorporation

- One Predict file object is created for each table or view.
The ID of the new file object consists of the creator name and name of the table or view joined by a hyphen.
- One field object is created for each field in the table or view.
The ID assigned to each new field object will be the name of the field in Adabas SQL. For each view that is incorporated, the appropriate parts of the subselect specification in the SQL statement CREATE will be copied to the file and field objects created.
- One Predict field object (type SP) for each composite index or key is created.
- One file relation object is created for each referential constraint.

Incorporation Option	
List incorporated code	Y the values of all attributes incorporated for the table or view are to be listed.

Incorporating Adabas Tables and Views in Batch Mode

Command: INCORPORATE ESQ

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
Table/View name	TABLE	1
Tables or Views	TYPE	2
Creator name	CREATOR	3
Connect automatic	AUTO-CONNECT	4
Function	SUB-FUNCTION	5
List incorporated code	LIST	6

Example

To incorporate the table 'TDEPT', code the command:

```
INCORPORATE ESQ  
TABLE=TDEPT,TYPE=T,,,SUB-FUNCTION=I
```

or

```
INCORPORATE ESQ  
TDEPT,T,,,I
```

The example above uses the Natural parameters ID=, and IA==

Incorporating Natural DDMs

Prerequisites and Restrictions

- Functions compare and connect functions are available both online and in batch mode; the incorporate subfunction is only available online.
- For DDMs of type DB2/ESQ, IMS or PROCESS, the functions I and V are not applicable.

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting DDMs

Display the Incorporate DDMs screen with function code I and object code DD in a Predict main menu or with the command INCORPORATE DDM.

```

10:00:00          ***** P R E D I C T  4.3.1  *****                2003-05-31
Plan    0          - Incorporate DDMs -

Current VM ..... HOME

DDM ID .....

Select option
  DDMs of type .....*

Processing option
  Connect automatic .. N (Y/N)
    
```

Parameters for Selecting DDMs to be Incorporated	
Current VM	If a Predict database object is found that is either linked to the Current VM and of type local or accessible from the network containing the Current VM, the master file incorporated from the DDM is linked to this database. A read-only field. If no current VM is specified in the session profile, the parameter Default current virtual machine in the Miscellaneous section of the General Defaults is used.
DDM ID	ID of the DDM. Asterisk notation is allowed.
DDMs of type	Limit the function to DDMs that access databases of the type specified. Enter an asterisk to display valid values.
Connect automatic	Y The DDM is connected to an existing the corresponding file object whenever the Test subfunction confirms that the two correspond.

DDMs which meet the given selection criteria and are not already connected to a Predict file object are then listed as follows.

```

13:13:32          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - Incorporate DDMs -

Search-crit.: DDM ID=ALL

Cmd DDM ID          DDM      Predict
                   type      file type      Remark

___ ACCOUNTING      PROCESS
___ BS2000-COMMAND  PROCESS
___ CATALOG          PROCESS
___ CATALOG-UPDATE  PROCESS
___ CHD-FORMATE1     ADABAS
___ CHD-TEST         ADABAS Conceptual file Diff. Filetype
    
```

Columns in the List	
DDM ID	ID of the DDM.
DDM type	The type of database that can be accessed by using the DDM with this ID: Adabas or VSAM or DB2 or IMS or Entire System Server.
Predict file type	The type of any data dictionary object for a file with this ID.
Remark	
Any of the following type-specific remarks or blank. For type-independent remarks see Remarks which apply to all Object Types.	
No related object	Applies to VSAM DDMs. The specified VSAM DDM cannot be incorporated as a data dictionary object until a data dictionary object for a related VSAM file has been created:
Different file type	Predict file object and entry in the NTDB differ.
No master DDM	Master DDM of logical VSAM view is missing.
No physical DDM	Master DDM of physical VSAM view is missing.
DBID is VSAM	Type of DDM different from corresponding entry in NTDB.
DBID is Adabas	Type of DDM different from corresponding entry in NTDB.
Super Natural DDM	Super Natural DDMs cannot be processed.

Incorporation Functions for DDMs

Valid functions: T, C, V and I.

Compare - Code T

Compares DDMs and corresponding Predict file objects. Differences between field lists and file attributes are reported.

If no differences are found and Connect automatic is set to Y, the DDM and the Predict file object are connected.

Connect - Code C

Connects DDMs to Predict file object and writes to the generation log of the Predict file object that a corresponding DDM existed when the connection was established.

Display Related DDMs - Code V

If the message No related object the IDs of the related DDM or DDMs that must be incorporated first can be displayed with the function V. The DDMs are listed in the order in which they have to be incorporated.

Incorporate - Code I

Incorporates information from the DDM to a new Predict file object of type A, F, L, R, U, T, V, W, 1 and 2 as appropriate and connects the DDM and the new Predict file object.

Prerequisites and Restrictions

- Before an object for a file of type W or L can be created, an object for the related file of type V must exist;
- Before an object for a file of type R can be created, an object for the related file of type L must exist.
- Before an object for a file of type U can be created, an object for the related file of type A must exist.
- Incorporate DDM can only be executed online.
- The V function (see above) can be used to identify related DDMs that must be incorporated first.

Scope of the Incorporation

- One Predict object of the type file will be created: The new file object has the DDM name as ID.
- The Predict file object is linked to a Predict database that is either linked to the Current VM and of type local or accessible from the network containing the Current VM.
- One field object will be created for each field of the DDM. Each new field object will be assigned the name of the field in the DDM.
Any prefix specified by the user and present in all field names will be removed to create the corresponding field IDs.
- One relationship object is created for each coupling definition.

Before the function is executed the following parameters can/must be specified for each file in an additional input screen

Incorporation Options													
Related master file	<p>ID of the Predict file object describing the file for which the specified DDM is a userview</p> <table border="1"> <thead> <tr> <th>Type</th> <th colspan="2">Type of Master File</th> </tr> </thead> <tbody> <tr> <td>L or W</td> <td>V</td> <td>Physical VSAM file</td> </tr> <tr> <td>R</td> <td>L</td> <td>Logical VSAM file</td> </tr> <tr> <td>U</td> <td>A</td> <td>Adabas file</td> </tr> </tbody> </table> <p>Leave this field blank if the DDM corresponds to a master file.</p>	Type	Type of Master File		L or W	V	Physical VSAM file	R	L	Logical VSAM file	U	A	Adabas file
Type	Type of Master File												
L or W	V	Physical VSAM file											
R	L	Logical VSAM file											
U	A	Adabas file											
Truncation prefix	If the names of all fields of the DDM have a common prefix and that prefix is specified here, it is removed from the field names to create the IDs of the field objects in the data dictionary.												
Comment lines	Y General comment lines in the DDM are incorporated as field descriptions with field type type **.												
Remark as abstract	Y Field comments in the DDM are to be incorporated.												
Old synonym mode	Y Fields with identical Adabas field names are regarded as synonyms if their type, format, length and descriptor options are the same. This switch can only be set if the Old mode synonyms in the Synonyms screen of the General Defaults menu is set to Y.												
List incorporated code	<p>Y the values of all attributes incorporated for the file or userview are to be listed.</p> <p>After the Incorporate DDM function has been executed, the function Edit elements of a file is executed and a CATalog is performed.</p> <p>If an error occurs, the editor is called so that the error can be corrected.</p> <p>Errors may occur, for example, because a DDM contains less information than an FDT about subdescriptors, superdescriptors and phonetic descriptors. Because errors are possible, this subfunction is not available in batch mode.</p>												

Incorporate DDM in Batch Mode

Command: INCORPORATE DDM

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
DDM ID	FILE-ID	1
DDMs of type	DDM-TYPE	2
Connect automatic	AUTO-CONNECT	3
Function	SUB-FUNCTION	4

Example

To connect the DDM with the name 'TEST-DDM' to the corresponding file object in Predict, code the command:

Incorporating Natural DDMs

Incorporate DDM in Batch Mode

```
INCORPORATE DDM  
FILE-ID=TEST-DDM , SUB-FUNCTION=C
```

or

```
INCORPORATE DDM  
TEST-DDM , , C
```

The example above uses the Natural parameters ID=, and IA==

Incorporating COBOL Copy Code

This function builds a Predict file and linked fields from COBOL copy code (record description or a subordinate section without a level 01 entry).

Up to 20 files can be built per session. The copy code may contain only the following:

- File definitions. Each line may contain a maximum of one data definition.
- Blank lines.
- Comment lines. Comments are added to the last valid file or field object that is incorporated.

It is not possible to incorporate information from COBOL programs.

The information incorporated is stored in Predict file objects of type sequential and field objects.

Prerequisites

The following resources are required for this function:

- Natural workfile 1 (record format F or FB with record length 80, V or VB)
- Natural workfile 2 (record format V or VB, record length at least 44).

In batch mode, the following additional resources are also required:

- Natural report 1
- Natural report 2
- Natural report 3.

Calling the Function

The Incorporate COBOL Copy Code screen is called with function code I and object type code CO in any Predict main menu or with command INCORPORATE COBOL.

```

09:59:05          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Incorporate COBOL Copy Code -

File ID .....

Incorporation options
Prefix ..... A-
Decimal character ..* P
Currency sign .....* $
Literal delimiter ..* '
List .....* R
Store ..... Y (Y,N)
Replace ..... N (Y,N)
Input type .....* W
Compiler .....* 7
Comment lines to ...* A
    
```

Parameters

File ID	<p>If file ID is not specified, level 01 entries on the workfile are interpreted as file IDs. A new file is created for each new level 01 entry. All other entries are interpreted as fields of these files.</p> <p>If file ID is specified, all entries on the workfile are interpreted as fields of this file. Level 01 entries are taken as groups.</p>
Incorporation Options	
Prefix	<p>Prefix to be added to any field ID with an invalid initial character for a Predict object ID. See Naming Conventions in the section General Information in the Predefined Object Types in Predict documentation.</p> <p>Field IDs that are too long are truncated from the right.</p>
Decimal character	<p>Determines how the input data on the workfile is interpreted. Default value is taken from generation defaults but can be overridden here.</p> <p>C Decimal comma. The DECIMAL-POINT IS COMMA clause is set in the SPECIAL-NAMES paragraph of the corresponding COBOL program.</p> <p>P Decimal point.</p>
Currency sign	<p>The currency symbol in the PICTURE clause. Any currency symbol valid in COBOL can be used. The default is dollar.</p>
Literal delimiter	<p>Literal delimiter character. Default value is taken from generation defaults but can be overridden here.</p> <p>" Double quotation mark. The compiler option QUOTE is used.</p> <p>' Single quotation mark. The compiler option APOST is used.</p>
List	<p>Determines which columns of each COBOL line are listed. Valid values:</p> <p>R Columns 73-80 on the right are not listed.</p> <p>L Columns 1-6 on the left are not listed.</p> <p>A All columns will be listed. Only valid in batch mode.</p>
Store	<p>Y The generated file description will be incorporated in Predict.</p> <p>N File description will be displayed on screen, but not incorporated in Predict.</p>

<p>Replace</p>	<p>Only valid if Store is set to Y. Valid values:</p> <p>Y Existing Predict file(s) and any dependent fields with same name will be replaced. Only files of type Conceptual, Other or Sequential will be replaced.</p> <p>N Error message will be given if file with same name exists, but dictionary objects will not be replaced. Default.</p>
<p>Input type</p>	<p>W Copy code is taken from Natural workfile 1.</p> <p>I Copy code is entered directly in the screen below. The last card must be END. For more information see your COBOL documentation.</p> <div data-bbox="331 846 1364 1214" style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p style="text-align: center;">ENTER ONE COBOL-LINE (FOR END ENTER 'END' FROM COLUMN 8):</p> <p style="text-align: center;">1 . . . 6 7 8 . . 12 . . . 2 3 4 5 6 72</p> </div> <p>Note: Sequence numbers are ignored.</p>
<p>Compiler</p>	<p>Determines the type of compiler used. Default value is taken from generation defaults but can be overridden here.</p> <p>7 Standard COBOL 74 compiler is used.</p> <p>8 COBOL 85 compiler is used.</p> <p>W COBOL compiler for WANG is used.</p>
<p>Comment lines to</p>	<p>Determines where comment lines in COBOL are written to.</p> <p>A Comments are written to the abstract of file/field object.</p> <p>D Comments are written to the extended description of file/field object.</p>

Incorporating COBOL Copy Code in Batch Mode

Command: INCORPORATE COBOL

Enter parameters on next line using positional or keyword form.

Field	Keyword	Position
File ID	FILE-ID	1
Prefix	PREFIX	2
Decimal character	DECIMAL-CHAR	3
Currency sign	CURRENCY-SIGN	4
List	LIST	5
Store	STORE	6
Replace	REPLACE	7
Input type	INPUT-TYPE	8
Literal delimiter	DELIMITER	9
Compiler	COMPILER	10
Comment lines to	COMMENT	11

Representation of COBOL Field Definitions in Predict

The table below shows how COBOL data definitions are documented in Predict after incorporating COBOL copy code.

COBOL USAGE Clause	COBOL PICTURE Clause	COBOL Number of digits/bytes		Predict Format Length		Meaning *	For COBOL Compiler
BINARY	none			I	2	binary	WANG
BINARY	P S V 9	1-4	2	I	2	binary	WANG
		5-9	4	I	4		
		10-18	8	I	8		
COMP	P S V 9	1-4	2	I	2	binary	
		5-9	4	I	4		
		10-18	8	I	8		
				P or PS	nn.m	packed numeric binary	WANG
COMP-1	not applicable			F	4	internal floating point short	
COMP-2	not applicable			F	8	internal floating point long	
COMP-3	P S V 9			P or PS	nn.m	packed numeric binary	
COMP-4	P S V 9	1-4	2	I	2	binary	
		5-9	4	I	4		
		10-18	8	I	8		
DISPLAY	AB			A	nnnnn	alphabetic	
DISPLAY	A * 9			A	nnnnn	alphanumeric	
DISPLAY	A * 9 B 0 /			A	nnnnn	alphanum. for print	
DISPLAY	A P V Z 9 0 / , . + - C R D B * \$			P or PS	nn.m	numerical for print	
DISPLAY	P S V 9			N or NS	nn.m	decimal external	
DISPLAY	E V 9 . + -			F	8		
DISPLAY	V 9 for mantissa	16		F	8	floating point external	
DISPLAY	99 for exponent	2		F	8		
INDEX	not applicable			not supported		Index	
PACKED-DECIMAL	P S V 9			P or PS	nn.m	packed numeric binary	85

* See the respective COBOL language description for more information on the different data formats.

Incorporating Super Natural Users

The Incorporate Super Natural user screen is displayed by selecting function code I and object code SU in a Predict main menu or by entering the direct command INCORPORATE SUPER.

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

```

10:01:45          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Incorporate NSP user -

User ID .....

Add user ..... N (Y/N)

Super Natural File
Database number .. 76
File number..... 252
    
```

Parameters	
User ID	ID of the Super Natural user to be processed. Asterisk notation is allowed.
Add user	Y Super Natural users not already defined as user objects in Predict will be added.
Super Natural File	
Database number	Number of the database where the Super Natural user description is stored.
File number	Number of the file where the Super Natural user description is stored.

Incorporating Super Natural User in Batch Mode

Command: INCORPORATE SUPER

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
User ID	USER-ID	1
Add user	ADD-USER	2
Database number	DBNR	3
File number	FNR	4

Example

To incorporate Super Natural users whose names starts with 'A' in batch, enter:

```

INCORPORATE SUPER
USER-ID=A* ,ADD-USER=Y
    
```

Incorporating Super Natural User in Batch Mode

Incorporating Super Natural Users

or

```
INCORPORATE SUPER  
A*, Y
```

The example above uses the Natural parameters IA== and ID=,

Incorporating Natural Security Users

Prerequisites and Restrictions

Only a Natural Security System Administrator can incorporate a Natural Security user.

Note:

See Concepts of Incorporation for basic information on how to use incorporation functions.

Selecting Natural Security Users

The Incorporate Natural Security user screen is displayed by selecting function code I and object code NS in a Predict main menu or by entering the direct command INCORPORATE SECURITY.

```

13:32:58          ***** P R E D I C T 4.3.1 *****          2003-05-31
                  - Incorporate Natural Security User -

User ID.....

Incorporation options
Add user..... N (Y/N)
From date..... 0000-00-00
User type.....*
with comments..... Y (Y/N)
with edit description N (Y/N)
    
```

Parameters	
User ID	ID of the Natural Security user to be processed. Asterisk notation is allowed.
Add user	Y Natural Security users that are not defined in Predict will be added to it.
From date	Limit the incorporation to user IDs which were added to the Natural Security system after the specified date.
User type	The type of user defined in Natural Security: A Administrator M Member P Person or blank any
with comments	Y User ID comments in the Natural Security system will be copied to Predict. Each comment line will be split and stored as two halves.
with edit description	Y User ID comments in the Natural Security system will be copied to the extended description.

Incorporating Natural Security Users in Batch Mode

Command: INCORPORATE Security

Enter parameters on next line in positional or keyword form.

Field	Keyword	Position
User	USER-ID	1
Add user	ADD-USER	2
from date	DATE	3
User type	TYPE	4
with comments	COMMENT	5
with edit description	DESC	6

Example

To incorporate Natural Security administrators whose names start with 'A', code the command:

```
INCORPORATE SECURITY
USER-ID=A* ,ADD-USER=Y ,TYPE=A
```

or in positional form:

```
INCORPORATE SECURITY
A* ,Y ,A
```

The example above uses the Natural parameters IA==, ID=, and IM=D

Incorporating Tables / Views of SQL Database Systems

Functional Scope

This function is available for the SQL systems listed below.

- Adabas D
- Ingres
- Informix
- Oracle
- Sybase

Note:

The screens and examples in this section are given for Adabas D but are also applicable to the other SQL systems.

Prerequisites

The prerequisites depend on whether you are using your SQL system locally or within a network environment:

- **Locally**
 - Entire Access must be installed
 - Specify the Entire Access database driver to be used with the dbms parameter in the NATCONF.CFG file. See your Entire Access documentation for more information.
- **Network environment**
 - Entire Access must be installed
 - Specify a remote connect string as described in your Entire Access documentation.

Restrictions

The following restrictions apply:

- **Ingres**

No rules or procedures are incorporated.
- **Adabas D**

When incorporating Adabas D views, the message "Master missing" does not appear even if the corresponding master file does not exist or is not accessible.
All constraints are incorporated as one check constraint of the file object in Predict.
- **Upper/Lower Case**

If the general default parameter Miscellaneous > Upper/lower case > Object ID is set to U and SQL tables and views with mixed-case names are to be incorporated, it is possible that some references are not recognized.

Selecting SQL Tables and Views

The respective Incorporate screen is called with function code I and the corresponding code in a Predict main menu or with the command INCORPORATE <sql-type>. See table below.

SQL System	Code	Command Word
Adabas D	BF	ESD-TABLE
Ingres	JF	INGRES-TABLE
Informix	XF	INFORMIX-TABLE
Oracle	OF	ORACLE-TABLE
Sybase	YF	SYBASE-TABLE

For example, the Incorporate Adabas D Tables/Views screen below is called with command INCORPORATE BF or INCORPORATE ESD-TABLE.

```

10:00:30          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Incorporate Adabas D Tables/Views -

Table / View name ....

Select options
  Tables or Views ...*
  Creator name .....

from DBID .....*
```

Parameters	
Table / View name	Name of table / view in the SQL environment. Asterisk notation can be used to specify a range of table/view names.
Select options	Determines whether tables or views are to be incorporated.
Tables or Views	T Tables V Views
Creator name	Limits the scope of the function to SQL tables/views with the specified creator.
From DBID	Number of the database as defined in the NATCONF.CFG file. This number identifies the SQL system from which tables/views are to be incorporated. Enter an asterisk to display a list of database numbers for selection. Note: If you are using more than one SQL system in a network environment, all possible databases are offered for selection, even if they are from another system.

Columns in the List	
Creator	Creator in the respective SQL system.
Table/view name	Name of the table/view.
T	Type of the SQL object: T SQL table V SQL view
Remark	
Either the following remark or blank.	
Master missing	The specified SQL view cannot be incorporated in Predict until a documentation object for a related SQL table or view has been created. Note: This remark is not given when incorporating Adabas D views.

Incorporation Functions for SQL Tables and Views

Subfunctions Incorporate and Display masters are available when incorporating SQL tables and views.

Display Masters - Code M

The SQL tables/views a view is related to must be documented in Predict before the SQL view can be incorporated. This subfunction finds all tables/views that must be documented in Predict before table/views can be incorporated.

Incorporate - Code I

Scope of the Incorporation

- One Predict file object is created for each table or view.
The ID of the new file object consists of the creator name and SQL name of the table or view joined by a hyphen.
- One field object is created for each field in the table or view.
The ID of the field is the name of the field in the respective SQL system. For each view that is incorporated, the appropriate parts of the subselect specification in the SQL statement CREATE will be copied to the file and field objects created.
- One Predict field object (type SP) is created for each composite unique constraint.
- One Predict field object (type SP) is created for each composite foreign key if there is not already a Predict field object for a composite unique constraint built from the same columns.
- One Predict file relation object is created for each foreign key.
- A verification is added and linked to the corresponding field for each check constraint that is defined for one column.
- Attributes of the corresponding file object are updated for each table check constraint (check expression for more than one column).
- One trigger object for each trigger that is connected to the file being incorporated. The trigger object is linked to the file object via association *Has TR*.

Before the function is executed, enter the following parameters in an additional input screen

Incorporation Options	
Comments as abstract	<p>Y String constants assigned with COMMENT ON are incorporated as the abstract of the corresponding Predict object:</p> <ul style="list-style-type: none"> ● Comments assigned to the table or view are to be copied to the file object. ● Comments assigned to a field of the table or view are to be copied to the field object. <p>Note: This parameter is not applicable to all SQL systems.</p>
List incorporated code	<p>Y The values of all attributes incorporated for the table or view are to be listed.</p>

Incorporate SQL Tables / Views in Batch Mode

Command: INCORPORATE <sql-type>

See list of codes and command words in the section Selecting SQL Tables and Views.

Enter the parameters on next line in positional or keyword form.

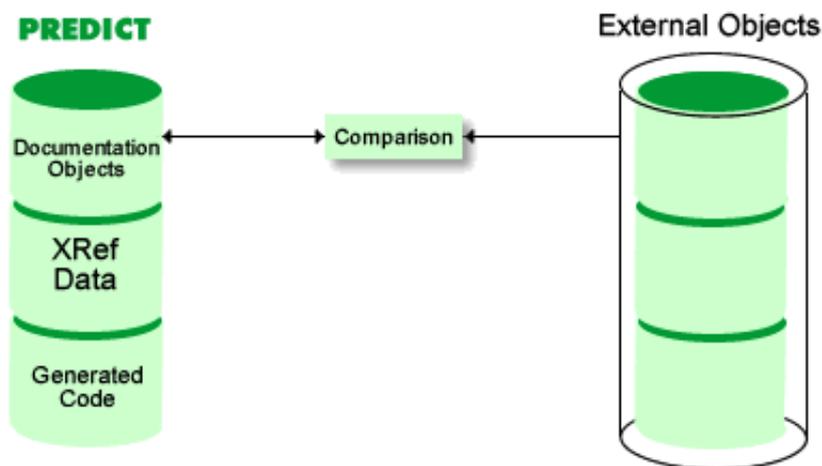
Field	Keyword	Position
Table/View name	TABLE	1
Tables or Views	TYPE	2
Creator name	CREATOR	3
From DBID	DBNR	4
Incorporate comments	COMMENT	5
List incorporated code	LIST	6

Note:

The keyword ABSTRACT can be used as a synonym for COMMENT.

Comparison

Comparison functions help to keep Predict documentation and the actual implementation of an information processing system consistent. Comparison functions are needed because the external objects can be changed with utilities of the programming environment independent of Predict, even if protection options for external objects are activated. See function General Defaults > Protection in the section Defaults in the **Predict Administration documentation**.



This section covers the following topics:

- Concepts of Comparison
- Calling Comparison Functions
- Comparing Different Types of Objects
- Comparing Adabas Databases
- Comparing Adabas Files
- Comparing Vista Translation Tables
- Comparing DDMs
- Comparing Adabas Tables/Views
- Comparing DB2 Databases
- Comparing DB2 Storagegroups
- Comparing DB2 Tablespaces
- Comparing DB2 Tables and Views

Concepts of Comparison

Functional Scope

The following Predict objects and external objects can be compared:

Predict Object	External Object
File	DDM
	Adabas File
	DB2 Tables and Views
	Adabas Tables and Views
All Adabas files and databases in a network	Vista Translation Table
Database	Adabas Database
	DB2 Database
Storagespace	DB2 Storagegroup
Dataspace	DB2 Tablespace
	SQL/DS Dataspace

If comparison functions find differences between an external object and its corresponding documentation object, either the implementation or the documentation can be changed to restore consistency.

Actions to restore consistency are either performed by the Predict comparison function or have to be executed manually.

The following actions can be executed by Predict Compare functions:

- **Update** documentation objects according to external objects.
- **Reset** the Modified after generation flag for documentation objects that do not differ from corresponding external objects.
See Resetting the Modified after generation Flag below.

The following actions must be executed manually:

- **Generate** a new external object from the documentation object.
- **Disconnect** the external object and the documentation object.
- **Purge** the external object or the documentation object.

The Save Set option helps to create a list of members which have to be processed according to changes in the documentation of an information processing system.

See Saving the Result of Comparison Functions in Sets.

Resetting the Modified after generation Flag

The Modified after generation flag of Predict objects indicates if a Predict object has been changed after an external object was generated from it.

With Predict file objects of type A, separate Modified after generation flags are set for changes to file attributes and changes to the field list of the file.

If compare functions find that a Modified after generation flag is set for a Predict object that does not differ from the external object, the flag is reset.

Modified after generation flags are reset if the respective check options of the compare function are activated:

- When comparing Adabas files, the Modified after generation flags for file attributes and for field lists can be reset separately.
Predict evaluates the check options File attributes and Fields.
- For all other objects types, only one Modified after generation flag is set.
All check options must set to Y if the flag is to be reset.

Format of Reports Created by Comparison Functions

When attributes of documentation and external objects differ, the attribute values of documentation objects are reported first and the information on external objects is indented and highlighted, together with a brief description of the difference.

```

13:40:33          ***** P R E D I C T 4.3.1 *****                2003-05-31
                   - Compare DD Objects with Adabas Files -

File ID ..... A-A                      PFnr ... 331
Database ID .. PREDICT-TEST-DB          PDBnr .. 188

File attributes
  Maxisn ..... 667
    Maxisn in database: 1289
  ISN reusage ..... Yes
    ISN reusage in database: No
  User ISN ..... Yes
    User ISN in database: No
    
```

Calling Comparison Functions

When working online, comparison functions are called from type-specific Compare screens called with code C and an external object code in a Predict main menu or with the corresponding command. See table in the section Calling Comparison Functions with Commands.

General Prerequisites for Comparison Functions

Predict objects and external objects to be compared must be connected. See the section Handling of External and documentation Objects in this documentation.

Note:

There is one exception: If Adabas Online Services are not installed, implemented Adabas files and Predict file objects of type A can be compared even if they are **not** connected.

A variety of Adabas attributes can only be compared if Adabas version 6 or above is used and Adabas Online Services are installed.

Selecting Objects for Comparison Functions

Comparison functions can either be applied to individual objects or groups of objects which meet selection criteria. The selection criteria vary for the different object types and are described in detail with the respective object type.

Check, List, Update and Save Options

When comparing external objects and Predict objects, Check, List, Update and Save options determine the scope of the operation:

- **Check Options**
Determine how attributes are to be compared.
If check options are not set, compare functions will not reset the Modified after generation flag. See Resetting the Modified after generation Flag.
- **List Options**
Determine the scope of results of the comparison to be displayed.
- **Update Options**
Determine which attributes of the Predict object are to be updated if differences were found.
- **Save Option**
See Saving the Result of Comparison Functions in Sets below.

The options available depend on the object type. All options are described in detail with the respective object type in the sections below.

Saving the Result of Comparison Functions in Sets

The IDs of members reported by comparison functions can be saved in sets for further processing. The following rules apply:

- To save the set of members, set the Save set option to Y before executing the function.
- With comparison functions, sets are saved in the library that contains the objects they describe. Hence, executing one Predict comparison function may save sets of objects in more than one library.
- The sets saved in a library can be maintained. See Operate on Sets in the section **LIST XREF for Natural** in the **Predict Reference documentation**.

Calling Comparison Functions with Commands

Comparison functions can be used both online and in batch mode.

The following table gives an overview of objects that can be compared, the codes to be entered in any Predict main menu and the corresponding command.

External Object Type	Code	Command
Adabas Database	AD	COMPARE ADA-DA
Adabas File	AF	COMPARE FDT
Vista Translation Table	AT	COMPARE VISTATAB
DDM	DD	COMPARE DDM
Adabas Table/View	EQ	COMPARE ESQ
DB2 Database	D2	COMPARE DB2-DATABASE
DB2 Storagegroup	SG	COMPARE STORAGEGROUP
DB2 Table/View	T2	COMPARE TABLE
DB2 Tablespace or SQL/DBspace	TS	COMPARE DB2-TABLESPACE

Comparing Different Types of Objects

The following external objects can be compared with Predict objects:

- Adabas Databases
- Adabas Files
- Vista Translation Tables
- DDMs
- Adabas Tables and Views
- DB2 Databases
- DB2 Storagegroups
- DB2 Tablespaces
- DB2 Tables and Views

Comparing Adabas Databases

Prerequisites

Adabas Online Services must be available if a database of Adabas Version 6 or above is to be compared.

If Natural Security function protection for Predict is activated, permission is required to compare Adabas databases.

See also General Prerequisites for Comparison Functions.

Calling the Function

The Compare DD Objects with Adabas DBs screen is called with code C and object code AD in any Predict main menu or with the command COMPARE ADABAS-DATABASE.

```

10:53:02          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Compare DD Objects with Adabas DBs -          Profile HNO

Current VM ..... HOME

Database ID .....
Database number .....
Belongs to VM .....

List options                                Update options
  DB attributes ....* D                      DB attributes ... N (Y/N)
  File list .....* D                          File list
                                           Unlink files ... N (Y/N)

Restrictions .....*   Profile HNO ,used

```

Parameters	
Current VM	ID of the virtual machine object identifying the current environment. A read-only field. This value can be changed in the Maintenance options of the Modify User Defaults function or with the command SET VM<virtual machine ID>. If no current VM is specified in the session profile, the Default current virtual machine parameter in the Miscellaneous section of the General Defaults is used.
Database ID	ID of the Predict database object to be compared. Asterisk notation can be used.
Database number	The number of the Adabas database to be compared.
Belongs to VM	Limits the function to databases linked to this virtual machine via association <i>Contains DA</i> . Asterisk notation can be used.
List options	
DB attributes	How attributes of the Predict database object are to be displayed. Y All attributes N No attributes D Different: only database attributes whose values differ.
File list	How files contained in the file list of the Predict database object are to be displayed: Y All files N No files D Different: only files that are either not implemented in the Adabas database or are not linked to a Predict documentation object.
Update options	
DB attributes	Y The attributes of the Predict database object are to be updated.
File list: Unlink files	Y Files that are not implemented in the Adabas database are to be removed from the file list of the Predict database object.

Comparing Adabas Databases in Batch Mode

Command: COMPARE ADABAS-DATABASE

Enter parameters on next line in the positional or keyword form.

Field	Keyword	Position
Database ID	DATABASE-ID	1
Database number	DBNR	2
Belongs to VM	PARENT	3
List options		
DB attributes	LIST-DB	4
File list	LIST-FILE	5
Update options		
DB attributes	UPD-DB	6
Unlink files	UNLINK-FILE	7

Example

To compare all databases of type 'A' with prefix 'TEST' and to display the file lists code the command:

```
COMPARE ADABAS-DATABASE
DATABASE-ID=TEST* ,LIST-FILE=Y
```

or

```
COMPARE ADABAS-DATABASE
TEST* , , , , Y
```

The above example uses the Natural parameter ID=, and IA==

Comparing Adabas Files

Prerequisites and Restrictions

- Adabas files and Predict files to be compared must be connected. There is one exception to this rule: if AOS is not installed, implemented Adabas files and Predict file objects can be compared even if they are not connected.
- If Natural Security function protection for Predict is activated, permission is required to compare Adabas files.

See also General Prerequisites for Comparison Functions.

Calling the Function

The Compare DD Objects with Adabas files screen is called with code C and object code AF in any Predict main menu or with the command COMPARE FDT.

```

10:55:35          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Compare DD Objects with Adabas Files -          Profile HNO

Current VM ..... HOME
File ID .....          File number .....
Contained in DA ..          Database number ..
Check options          List options
  File attributes ..... Y (Y/N)  Files .....* D
  Fields ..... Y (Y/N)          Fields .....* D
  Even if only FDT exists .. Y (Y/N)  Members using differences ..* N
  Adabas password .....
Update options
  File attributes .....* N
  Field sequences ..... N (Y/N)
  Add fields ..... N (Y/N)
  Purge fields ..... N (Y/N)
  Field attributes .....* N

Restrictions .....*  Profile HNO ,used          Save set .. N (Y/N)

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi Selfi Prof Main
    
```

Parameters	
Current VM	ID of Predict virtual machine object identifying the environment of the Adabas file. A read-only field. See also table with Parameter descriptions in the section Comparing Adabas Databases .
File ID	ID of the Predict file object of type A to be compared with the connected Adabas file. Asterisk notation can be used.
Contained in DA	Limits the function to files in this database. Asterisk notation can be used.
File number	File with this file number is processed.
Database number	Only file in database with this number is processed.

Check options	
If the check options File attributes or Fields are not set, compare functions will not reset the Modified after generation flag. See Resetting the Modified after generation Flag for more information.	
File attributes	<p>Y Attributes of the file are to be compared with the implementation of the file in the database (space allocation, for example). Attributes not required when loading a file are compared only if they are specified in the Predict object. This setting is valid only when Adabas Online Services is available and the file that is to be compared resides in a database of Adabas Version 6 or above.</p>
Fields	<p>Y The field list of the file object is to be compared with the Adabas file definition table (FDT).</p>
Even if only FDT exists	<p>Y A file is to be compared even if it has been deleted with the Keep FDT option.</p> <p>N File is not compared. For this setting, three conditions must be met:</p> <ul style="list-style-type: none"> ● the Adabas Online Services selectable unit must be installed ● the file to be compared must reside in a database of Adabas Version 6 or above ● the file to be compared must not have been deleted with the Keep FDT option.
Adabas password	Password required to access the file if the file is protected by Adabas security. Passwords are not converted to upper case.
List options	
Files	<p>How attributes of the Predict file object are to be displayed.</p> <p>Y All file attributes</p> <p>N No file attributes</p> <p>D Different: only file attributes whose Predict values differ from the implementation.</p>
Fields	<p>How fields linked to the Predict file object are to be displayed.</p> <p>Y All fields</p> <p>N No fields</p> <p>D Different: only fields whose Predict definitions differ from the FDT.</p>

<p>Members using diff.</p>	<p>Determines the amount of information to be displayed on members using external objects that differ from the corresponding Predict objects.</p> <p>The following differences are possible:</p> <ul style="list-style-type: none"> ● file numbers of Predict file objects differ from the FDT or a userview of such a file. ● Predict field objects differ from the FDT. <p>Valid values:</p> <p>Y Display types and names of all members using files that differ in any of the above ways.</p> <p>N Do not display members.</p> <p>C Count the number of members per library.</p>
<p>Update options</p>	
<p>Update options are effective only if parameters Save set and Programs using differences are set to N.</p>	
<p>File attributes</p>	<p>How attributes of the Predict file object are to be updated. Provided that the Adabas Online Services selectable unit is available and the file that was compared resides in a database of Adabas Version 6 or above, one of the following can be specified:</p> <p>R File attributes required when loading a file and any other file attributes specified in the Predict object</p> <p>N No file attributes</p> <p>O File attributes not required when loading a file are updated only if they are specified in the Predict object.</p>
<p>Field sequence</p>	<p>Y The field list is to be rearranged to match the sequence of the fields in the FDT.</p>
<p>Add fields</p>	<p>Y Fields in the FDT that are missing in the field list are to be inserted in the field list.</p>
<p>Purge fields</p>	<p>Y Fields in the field list that are missing from the FDT are to be removed from the field list.</p>

Field attributes	<p>How attributes of the fields linked to the Predict file object are to be changed to match the FDT.</p> <p>A All field attributes</p> <p>N No field attributes</p> <p>I Field attributes that can only be changed in the FDT by deleting and re-implementing the file.</p> <p>After fields have been updated, the function Edit elements of a file is executed automatically and a catalog is performed.</p> <p>Note: We recommend that you set all or none of the field update parameters to Y. Otherwise confusing results may occur, especially if the Field sequence parameter is not set to Y. For example, a periodic group may be inserted out of sequence. A cursory inspection might then give the false impression that the group's member fields are on the wrong level.</p>
Save set	<p>Y Program types and names reported by the function are saved in a set. See also Saving the Result of Comparison Funtions in Sets.</p>

Comparing Adabas Files in Batch Mode

Command: COMPARE FDT

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
File ID	FILE-ID	1
Contained in DA	DB	2
File number	FNR	3
Database number	DBNR	4
Check options		
File attributes	CHECK-FILE	5
Fields	CHECK-FIELD	6
Even if only FDT exists	EVEN-ONLY-FDT	7
Adabas password	PSW	8
List options		
Files	LIST-FILE	9
Fields	LIST-FIELD	10
Members using differences	LIST-PROGRAM	11
Save set	SAVE-SET	12
Update options		
File attributes	UPD-FILE	13
Field sequence	UPD-SEQUENCE	14
Add fields	ADD-FIELD	15
Purge fields	PURGE-FIELD	16
Field attributes	UPD-FIELD	17

Example

To compare all files in database 'TEST' and display all fields with a different definition, code the command:

```
COMPARE FDT
DB=TEST, LIST-FIELD=D
```

or

```
COMPARE FDT
,TEST,,,,,,,,,D
```

The example above uses the Natural parameter ID=, and IA==

Comparing Vista Translation Tables

Prerequisites

Adabas Vista must be installed. See also General Prerequisites for Comparison Functions.

Note:

To compare a Vista translation table and Predict Vista elements, external and documentation objects do not have to be connected.

Calling the Function

The Compare DD Objects with Vista screen is called with code C and object code AT in any Predict main menu or with the command COMPARE VISTATAB.

```

13:32:48          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Compare DD with Vista table -          Profile HNO

Current VM ..... HOME

List Vista elements .....* D
    
```

Parameters	
Current VM	ID of Predict virtual machine object identifying the current environment. A read-only field. See table of Parameter descriptions in the section Comparing Adabas Databases for detailed information.
List Vista elements	The amount of information to be displayed on entries in an Vista translation table. Y All entries in the Vista translation table N No entries D Only entries in Vista tables differing from corresponding Predict Vista elements.

Comparing Vista Translation Tables in Batch Mode

Command: COMPARE VISTATAB

Enter parameter in next line.

Field	Keyword
List Vista elements	LIST

Example

To compare an Vista translation table and list all Vista elements, code the command:

```
COMPARE VISTATAB  
LIST=Y
```

or

```
COMPARE VISTATAB  
Y
```

The above example uses the Natural parameter ID=, and IA==

Comparing DDMs

Prerequisites

See also General Prerequisites for Comparison Functions.

If Natural Security is installed and the file (DDM) has been defined to the Security system, only administrators may compare the file. Also, if the file object in Natural Security has an Owner list, the DDM may be compared only by one of its owners. No countersignatures are necessary. A file is not selected if a DDM connected to the file is defined to the Security system and the user is either not an administrator or not in the Owner list.

Calling the Function

The Compare DD Objects with DDMs screen is called with code C and object code DD in any Predict main menu or with the command COMPARE DDM.

```

10:56:12          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Compare DD Objects with DDMs -          Profile HNO

Current VM .... HOME

File ID .....          Files of type ..*
Contained in DA ...          File number .....

Check options          List options
File attributes .... Y (Y/N)    Fields .....* D
Fields ..... Y (Y/N)          Verifications .....* D
Verifications ..... Y (Y/N)    Members using differences ....* N
Adabas Version ....* I7

Restrictions .....*    Profile HNO ,used          Save set ..... N (Y/N)
    
```

Parameters	
Current VM	ID of the virtual machine object identifying the current environment. A read-only field. See also table with Parameter descriptions in the section Comparing Adabas Databases .
File ID	The ID of the Predict file object or objects which are to be compared with the connected DDMs of userviews or files. Asterisk notation can be used.
Files of type	Limit the function to files of the specified type.
File number	File with this file number is processed.
Contained in DA	Limit the function to files in the specified database.
Check options	
If check options are not set, compare functions will not reset the Modified after generation flag. See Resetting the Modified after generation Flag for more information.	
File attributes	Y Attributes of the file are to be compared with the implementation of the file in the DDM and any differences in file number, database ID and file type-specific attributes are to be displayed.

Fields	<p>Y The field list of the file object is to be compared with the DDM.</p>
Verifications	<p>Y The verifications of the fields are to be compared with the DDM to find any that</p> <ul style="list-style-type: none"> ● have incompatible formats, ● are not activated, or ● have been modified, disconnected or resequenced since the DDM was generated.
Adabas version	<p>The version of Adabas for which the DDM was generated. This determines the representation of integers and floating point numbers, hyperdescriptors and the size of the DBnr and Fnr in the DDM.</p>
List options	
Fields	<p>How fields linked to the Predict file object are to be displayed.</p> <p>Y All fields</p> <p>N No fields</p> <p>D Different: only fields whose Predict definitions differ from the DDM.</p>
Verifications	<p>How verifications of the fields are to be displayed.</p> <p>Y All verifications</p> <p>N No verifications</p> <p>D Different: only verifications that are not activated or have been modified, disconnected or resequenced since the DDM was generated.</p>
Members using diff.	<p>Determines the amount of information to be displayed on members using external objects that differ from the corresponding Predict objects:</p> <p>The following differences are possible:</p> <ul style="list-style-type: none"> ● A verification has been modified, disconnected or resequenced since the DDM was generated. ● A Predict field definition differs from the DDM. ● The file number of a Predict file object differs from the DDM or a user view of such a file. <p>Valid values:</p> <p>Y Display types and names of all members using objects that differ in any of the above ways.</p> <p>N Do not display members.</p> <p>C Count the number of members per library.</p>

Save set	Y Program types and names reported by the function are saved in a set. See also Saving the Result of Comparison Funtions in Sets.
----------	---

Comparing DDMs in Batch Mode

Command COMPARE DDM

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
File ID	FILE-ID	1
Files of type	TYPE	2
File number	FNR	3
Contained in DA	DB	4
Check options		
File attributes	CHECK-FILE	5
Fields	CHECK-FIELD	6
Verifications	CHECK-RULE	7
Adabas version	ADA-VER	8
List options		
Fields	LIST-FIELD	9
Verifications	LIST-RULE	10
Members using differences	LIST-PROGRAM	11
Save set	SAVE-SET	12

Example

To compare all files that begin with 'TEST' and to display members using differences code the command:

```
COMPARE DDM
FILE-ID=TEST* , LIST-PROGRAM=Y
```

or

```
COMPARE DDM
TEST* , , , , , , , , Y
```

The example above uses the Natural parameter ID=, and IA==

Comparing Adabas Tables/Views

Prerequisites

Parts of the ADVANCED Interactive Facilities of Adabas SQL Server must be installed within Natural. For detailed information see the requirements table in the respective part of the **Predict Installation documentation**.

Calling the Function

The Compare DD Objects with Adabas Tables/Views screen is called with code C and object code EQ in any Predict main menu or with the command COMPARE ESQ.

```

13:01:20          ***** P R E D I C T  4.3.1  *****                2003-05-31
Plan    0          - Compare DD with Adabas Tbls/Views -                Profile HNO

Current VM ..... HOME
File ID .....                               Files of type ..*
Contained in DA ...

Check options                                List options
File attributes .... Y (Y/N)                 Files .....* D
Fields ..... Y (Y/N)                        Fields .....* D
                                           Members using differences ....* N

Update options
File attributes .... N (Y/N)
Field sequences .... N (Y/N)
Add fields ..... N (Y/N)
Purge fields ..... N (Y/N)
Field attributes ... N (Y/N)

Restrictions .....*   Profile HNO ,used           Save set ... N (Y/N)

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
    
```

Parameters	
Current VM	ID of the virtual machine object identifying the current environment. A read-only field. See table with Parameter descriptions in the section Comparing Adabas Databases for more information.
File ID	The ID of the Predict file object(s) to be compared with the connected Adabas tables or views. Asterisk notation can be used.
Files of type	For Adabas tables: must be type A (with SQL usage set to Y) or type AT. For Adabas views: must be type B.
Contained in DA	Limit the function to tables and views which reside in the specified database of type A.
Check options	
If check options are not set, compare functions will not reset the Modified after generation flag. See Resetting the Modified after generation Flag for more information.	

File attributes	Y Attributes of the file are to be compared with the implementation of the file in the Adabas table/view.
Fields	Y The field list of the file object is to be compared with the Adabas table/view.
List options	
Files	How attributes of the Predict file object are to be displayed. Y All file attributes N No file attributes D Different: only file attributes whose Predict values differ from the implementation.
Fields	How fields linked to the Predict file object are to be displayed. Y All fields N No fields D Different: only fields whose Predict definitions differ from the Adabas table/view.
Members using diff.	Determines the amount of information to be displayed on members using fields that have been compared and whose Predict definition differs from the Adabas table/view Y Display the names and types of all member. N Do not display members. C Count the number of members per library.
Update options	
File attributes	Y File attributes are to be updated.
Field sequence	Y The field list is to be rearranged to match the sequence of the fields in the Adabas table/view.
Add fields	Y Fields in the Adabas table/view missing in the field list are to be inserted in the field list.
Purge fields	Y Fields that are missing in the Adabas table/view are to be removed from the field list.
Field attributes	Y Attributes of the fields linked to the Predict file object are to be changed to match the Adabas table/view.

Save set	Y Program types and names reported by the function are saved in a set. See also Saving the Result of Comparison Functions in Sets.
----------	--

Comparing Adabas Tables and Views in Batch Mode

Command: COMPARE ESQ

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
File ID	FILE-ID	1
Files of type	TYPE	2
Contained in DA	DB	3
Check options		
File attributes	CHECK-FILE	4
Fields	CHECK-FIELD	5
List options		
Files	LIST-FILE	6
Fields	LIST-FIELD	7
Members using differences	LIST-PROGRAM	8
Save set	SAVE-SET	9
Update options		
File attributes	UPD-FILE	10
Field sequences	UPD-SEQUENCE	11
Add fields	ADD-FIELD	12
Purge fields	PURGE-FIELD	13
Fields attributes	UPD-FIELD	14

Example

To compare all files of type A or B with prefix 'TEST' and display all fields with a different definition, code the command:

```
COMPARE ESQ
FILE-ID=TEST* , LIST-FIELD=D
```

or

```
COMPARE ESQ
TEST* , , , , D
```

Example

Comparing Adabas Tables/Views

The example above uses the Natural parameter ID=, and IA==

Comparing DB2 Databases

Prerequisites

DB2 must be installed. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation**

See also General Prerequisites for Comparison Functions.

Calling the Function

The Compare DD Objects with DB2 DBs screen is called with code C and object code D2 in any Predict main menu or with the command COMPARE D2.

```

09:42:13          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Compare DD Objects with DB2 DBs -          Profile HNO

Database ID .....
Belongs to VM .....

List options                                Update options
DB attributes .....* D                      DB attributes ..... N (Y/N)
Dataspaces list ....* D                     Links
File list .....* D                          Unlink dataspace ... N (Y/N)
                                           Unlink files ..... N (Y/N)

Restrictions .....* Profile HNO ,used
    
```

Parameters	
Selection criteria	
Database ID	The ID of the Predict database object or objects of type D which are to be compared with the connected DB2 database or databases. Asterisk notation can be used.
Belongs to VM	Limits the scope of the function to DB2 databases linked to this virtual machine via association <i>Contains DA</i> . Asterisk notation can be used.
List options	
DB attributes	How attributes of the Predict database object are to be displayed. Y All database attributes N No database attributes D Different: only database attributes whose Predict values differ from the implementation.

Dataspace list	How Predict dataspace objects linked to the Predict database object via association <i>Contains DC</i> are to be displayed. Y All dataspaces N No dataspaces D Different: only dataspaces that are not implemented as DB2 tablespaces and DB2 tablespaces that are not documented with Predict dataspace are contained in the list.
File list	How files linked to the Predict database object are to be displayed. Y All files N No files D Different: only files that are either not implemented in the DB2 database or not defined in Predict.
Update options	
DB attributes	Y The attributes of the Predict database object are to be updated.
Unlink dataspace	Y Dataspace not implemented as DB2 tablespaces are to be removed from the dataspace list of the Predict database object.
Unlink files	Y Files that are not implemented in the DB2 database are to be removed from the file list of the Predict database object. Note: Unlink files can only be set to Y if Unlink dataspace is also set to Y.

Comparing DB2 Databases in Batch Mode

Command: COMPARE DB2

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
Database ID	DATABASE-ID	1
Belongs to VM	PARENT	2
List options		
DB attributes	LIST-DB	3
Dataspace list	LIST-DS	4
File list	LIST-FILE	5
Update options		
DB attributes	UPD-DB	6
Unlink dataspace	UNLINK-DS	7
Unlink files	UNLINK-FILE	8

Example

To compare all databases of type D with prefix 'TEST' and to display the file lists code the command:

```
COMPARE DB2
DATABASE-ID=TEST* ,LIST-FILE=Y
```

or

```
COMPARE DB2
TEST* , , , Y
```

The above example uses the Natural parameter ID=, and IA==

Comparing DB2 Storagegroups

Prerequisites

DB2 must be installed. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation**

See also General Prerequisites for Comparison Functions.

Calling the Function

The Compare DD Objects with Stogroups screen is called with code C and object code SG in any Predict main menu or with the command COMPARE STORAGEEGROUP.

```

09:40:44          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Compare DD Objects with Stogroups -          Profile HNO

Storagespace ID ...

Update option          List option
  Storagespace attributes .. N (Y/N)          Storagespace attributes ....* D

Restrictions .....*          Profile HNO ,used
    
```

Parameters	
Selection criteria	
Storagespace ID	ID of the Predict storagespace object to be compared with a DB2 storagegroup. Asterisk notation can be used
List option	
Storagespace attributes	How attributes of the Predict storagespace object are to be displayed. Y Attributes of the Predict storagespace that differ from the DB2 storagegroup are listed. N No attributes are listed. D Different: only attributes of storagespace that are different from the storagegroup are listed.
Update option	
Storagespace attributes	Y Attributes of the Predict storagespace that differ from the DB2 storagegroup are updated.

Comparing DB2 Storagegroups in Batch Mode

Command: COMPARE STORAGEGROUP

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
Storagespace ID	STOSPACE-ID	1
List option		
Storagespace attributes	LIST-ST	2
Update options		
Storagespace attributes	UPD-ST	3

Example

To compare all storagespaces with prefix 'TEST' and list all attributes, code the command:

```
COMPARE STORAGEGROUP
STOSPACE-ID=TEST* ,LIST-ST=Y
```

or

```
COMPARE STORAGEGROUP
TEST* ,Y
```

The example above uses the Natural parameter ID=, and IA==

Comparing DB2 Tablespaces

Prerequisites

DB2 must be installed. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation**

See also General Prerequisites for Comparison Functions.

Calling the Function

The Compare DD Objects with Tablespaces screen is called with code C and object code TS in any Predict main menu or with the command COMPARE TABLESPACE

```

09:44:54          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Compare DD Objects with Tablespaces -          Profile HNO

Dataspace ID .....
Located in DA .....

Check options                                List options
  Dataspace attributes ... Y (Y/N)          Dataspace attributes .....* D
  Files ..... Y (Y/N)                      Files .....* D

Update options
  Dataspace attributes ... N (Y/N)
  Unlink files ..... N (Y/N)

Restrictions .....* Profile HNO ,used

```

Parameters	
Dataspace ID	ID of the Predict dataspace object that is to be compared. Asterisk notation can be used.
Contained in DA	Predict database object the dataspace object is linked to (via association <i>Contains DC</i>). Asterisk notation can be used.
Check options	
If check options are not set, compare functions will not reset the Modified after generation flag. See Resetting the Modified after generation Flag for more information.	
Dataspace attributes	Y Attributes of the Predict dataspace are compared with the DB2 tablespace.
Files	Y The file list of the dataspace is to be compared with the DB2 tablespace.
List options	
Dataspace attributes	How dataspace attributes are reported. Y All attributes of the Predict dataspace are reported. N No attributes are reported. D Different: only attributes of dataspace that are different from the tablespace are reported.
Files	How files linked to the dataspace via association <i>Contains FI</i> are to be reported: Y All files N No files D Different: only files that are either not implemented in the DB2 tablespace or not defined in Predict.
Update options	
Dataspace attributes	Y Attributes of the Predict dataspace that differ from the DB2 tablespace are changed.
Unlink files	Y Files that are not implemented in the DB2 tablespace are removed from the file lists of the Predict dataspace object and the database objects the dataspace belongs to.

Comparing DB2 Tablespaces in Batch Mode

Command: COMPARE TABLESPACE

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
Dataspace ID	DATASPACE-ID	1
Contained in DA	DB	2
Check options		
Dataspace attributes	CHECK-DS	3
Files	CHECK-FILE	4
List options		
Dataspace attributes	LIST-DS	5
Files	LIST-FILE	6
Update options		
Dataspace attributes	UPD-DS	7
Unlink files	UNLINK-FILE	8

Example

To compare all tablespaces with prefix 'TEST' and display all files, code the command:

```
COMPARE TABLESPACE
DATASPACE-ID=TEST* , LIST-FILE=Y
```

or

```
COMPARE TABLESPACE
TEST* , , Y
```

The example above uses the Natural parameter ID=, and IA==

Comparing DB2 Tables and Views

Prerequisites and Restrictions

DB2 must be installed. See the section DB2 and SQL/DS in the **Predict and Other Systems documentation**

See also General Prerequisites for Comparison Functions.

Calling the Function

The Compare DD Objects with DB2 files screen is called with code C and object code T2 in any Predict main menu or with the command COMPARE TABLE.

```

09:44:12          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          - Compare DD Objects with DB2 Files -          Profile HNO

File ID .....          Files of type ..*
Contained in DA ...

Check options          List options
File attributes .... Y (Y/N)    Files .....* D
Fields ..... Y (Y/N)          Fields .....* D
                                Members using differences .....* N

Update options
File attributes .... N (Y/N)
Field sequences .... N (Y/N)
Add fields ..... N (Y/N)
Purge fields ..... N (Y/N)
Field attributes ... N (Y/N)

Restrictions .....*    Profile HNO ,used          Save set ..... N (Y/N)

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJl Flip Print Impl AdmFi SelfI Prof Main
    
```

Parameters	
Selection criteria	
File ID	The ID of the Predict file object or objects which are to be compared with the connected DB2 tables or views. Asterisk notation can be used.
Files of type	Limit the function to files of the specified type.
Contained in DA	Limit the function to tables and views which reside in the specified database of type D. Asterisk notation can be used.
Check options	
If check options are not set, compare functions will not reset the Modified after generation flag. See Resetting the Modified after generation Flag for more information.	
File attributes	Y Attributes of the file are to be compared with the implementation of the file in DB2.

Fields	Y The field list of the file object is to be compared with the DB2 table or view.
List options	
Files	How attributes of the Predict file object are to be displayed. Y All file attributes N No file attributes D Different: only file attributes whose Predict values differ from the implementation.
Fields	How fields linked to the Predict file object are to be displayed. Y All fields N No fields D Different: only fields whose Predict definitions differ from the DB2 table or view.
Members using diff.	Determines the amount of information to be displayed on members using fields that have been compared and whose Predict definition differs from the DB2 table or view Y Display the names and types of all member. N Do not display members. C Count the number of members per library.
Update options	
File attributes	Y Attributes of the Predict file object and, for files of type E, the subselect specification of the SQL create statement are to be updated. The triggers linked to the file via association <i>Has TR</i> are also updated.
Field sequence	Y The field list is to be rearranged to match the sequence of the fields in the DB2 table or view.
Add fields	Y Fields in the DB2 table or view missing in the field list are to be inserted in the field list. Distinct types not already documented in standard file SAG-DISTINCT-TYPE will be added as fields. The connection between the standard field and the table field will be established.
Purge fields	Y Fields that are missing in the DB2 table or view are to be removed from the field list.

Field attributes	<p>Y Attributes of the fields linked to the Predict file object are to be changed to match the DB2 table or view.</p> <p>If a field has a connection to a standard field of SAG-DISTINCT-TYPE and the type definition differs from the DB2 catalog, the field attributes will be updated and the field will be marked non-standard.</p> <p>For files of type E, the attributes that are to be updated include the subselect specification of the SQL create statement for each field.</p> <p>Comments of fields are not updated.</p>
Save set	<p>Y Program types and names reported by the function are saved in a set. See also Saving the Result of Comparison Functions in Sets.</p>

Comparing DB2 Tables and Views in Batch Mode

Command: COMPARE TABLE

Enter parameters on the next line in positional or keyword form.

Field	Keyword	Position
File ID	FILE-ID	1
Files of type	TYPE	2
Contained in DA	DB	3
Check options		
File attributes	CHECK-FILE	4
Fields	CHECK-FIELD	5
List options		
Files	LIST-FILE	6
Fields	LIST-FIELD	7
Members using differences	LIST-PROGRAM	8
Save set	SAVE-SET	9
Update options		
File attributes	UPD-FILE	10
Field sequences	UPD-SEQUENCE	11
Add fields	ADD-FIELD	12
Purge fields	PURGE-FIELD	13
Fields attributes	UPD-FIELD	14

Example

To compare all files of type D or E with prefix 'TEST' and display all fields with a different definition, code the command:

```
COMPARE TABLE  
FILE-ID=TEST* ,LIST-FIELD=D
```

or

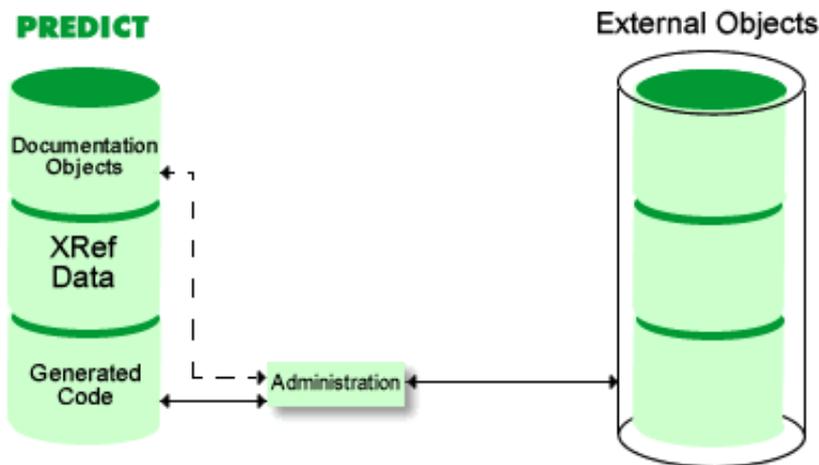
```
COMPARE TABLE  
TEST* , , , , D
```

The example above uses the Natural parameter ID=, and IA==

Administration of External Objects

Information stored in Predict objects can be used to generate external objects, and documentation objects can be incorporated from external objects. External objects generated from documentation objects and documentation objects incorporated from external objects are connected. Predict provides functions to administrate external objects connected to documentation objects. These functions are described in this section.

See the section Handling of External and Documentation Objects in this documentation for a more detailed description of how external and Predict objects are handled.



This section covers the following topics:

- Overview of Options
- Administrating Different Types of External Objects
- Databases
- Dataspaces
- Files
- Programs
- Stagespaces

Overview of Options

Overview of Administration Functions

Administration functions are used to perform the following tasks. See External Object Types and Administration Functions for a list of external objects and the functions that can be applied.

- **Disconnect implementation**
Disconnects external objects from their corresponding Predict objects. The implementation pointer from the documentation object to the external object is deleted, but the objects themselves remain intact.

Command: DISCONNECT <ext.object-type><dict.object-ID>[<parameters>]

- **Purge implementation**

Purges external objects and any dependent objects are physically deleted. It is sometimes necessary to delete an external object before the connected Predict object can be deleted.

Command: PURGE <ext.object-type> <dict.object-ID> [<parameters>]

- **Display implementation**

Displays external objects. The following information is displayed:

- Predict documentation data: object ID, type, creation and modification dates.
- Entire System Server data if external object was stored with Entire System Server.
- Generation options: the most important generation options are listed.
- Generated code.

Command: DISPLAY <ext.object-type> <dict.object-ID> [<parameters>]

- **Select implementation**

Selects external objects for further processing.

Command: SELECT <ext.object-type> <dict.object-ID> [<parameters>]

- **Rename implementation**

Moves generated code to another member and/or to another library.
Only applicable to copy code or utility cards generated from files.

Command: RENAME <ext.object-type> <dict.object-ID> [<parameters>]

Note:

This command cannot be used in batch mode. If the command is used online, not all parameters can be specified and must be supplied in the screens that appear.

- **Refresh file**

Deletes all records stored in Adabas files or DB2 tables/views.

Command: REFRESH <ext.object-type> <dict.object-ID> [<parameters>]

- **Purge Vista elements**

Deletes entries in Vista translation tables. This function cannot be executed in batch mode.

Command: PURGE VISTATAB

General Rules

- External objects to be processed are identified by the Predict documentation object to which they are connected.
- For all functions except Select implementation, an external object type must be specified.
- The standard functions Disconnect implementation and Purge implementation and the file-specific functions Rename implementation and Refresh file can only be applied to one object at a time.
- If the Predict object was incorporated or connected with an Incorporation function, no generation protocol is stored in Predict.

Commands

All functions except Rename implementation and Purge VISTATAB can be executed with direct commands. See Overview of Administration Functions for syntax.

Batch Mode

All Select functions for administrating implemented objects can be executed in batch mode.

Command: SELECT <ext.obj.type> <dict.object-ID> [<parameters>]

See list of external object types. The parameters are listed in the respective sections of this section.

This command produces a report.

Administrating External Objects Stored with Entire System Server

With Entire System Server, generated code for COBOL, PL/I, Assembler (BAL), FORTRAN, C, ADACMP/ADAWAN, ADAINV, Adabas Security, or Adabas VSAM BRIDGE can be stored directly as a member in an operating system library on a local or remote machine.

If an external object was stored with Entire System Server, the administration functions Display implementation, Purge implementation and Rename implementation list the following Entire System Server information:

- database ID
- DSname
- volser
- operating system member.

Purging Objects Stored with Entire System Server

Please note the following when applying the Purge implementation function to objects stored with Entire System Server:

DELETE

Generated code stored in the Predict system file is purged.

SCRATCH

Generated code stored in the Predict system file and operating system member is purged.

External Object Types and Administration Functions

This table lists all types of external objects and the administration functions that can be applied to them.

External Object	Code	Discon.	Display	Purge	Select	Rename	Refresh file	Purge VISTATAB
Connected to Predict Databases								
Adabas database	AD	Y						Y
IMS or DL/1	ND	Y						
DB2 database	D2	Y	Y	Y(1)				
Connected to Predict Files								
ADACMP/ ADAWAN	AC		Y	Y	Y	Y		
Adabas file	AF	Y		Y	Y		Y	
ADAINV cards	AI		Y	Y	Y	Y		
Vista Trans. table	AN				Y(2)			Y
ADASCR	AS		Y	Y	Y	Y		
Adabas-VSAM	AV		Y	Y	Y	Y		
BAL/Assembler	BA		Y	Y	Y	Y		
COBOL	CO		Y	Y	Y	Y		
SQL CREATE Statement	CR		Y	Y	Y	Y		
DDM for Natural	DD	Y	Y	Y	Y			
NSC file	NF			Y	Y	Y		
NSP file	SF	Y		Y	Y			
Preprocessor for 3GL	DD	Y		Y	Y			
DB2 table/view	T2	Y	Y	Y	Y	Y	Y	
Adabas table/view	EQ	Y	Y	Y	Y	Y		
FORTRAN	FO		Y	Y	Y	Y		
Language C	CC		Y	Y	Y	Y		
PL/I	PL		Y	Y	Y	Y		
Verification rule	RU		Y		Y			
UDF for DL/1	UD	Y		Y(3)	Y			
Connected to Predict Dataspaces								
DB2 tablespace	TS	Y	Y	Y	Y			
Connected to Predict Programs								
DB2 procedure/function	P2	Y	Y	Y	Y			
Connected to Predict Storagespaces								
DB2 storagespace	SG	Y	Y	Y	Y			

(1) A DB2 database can only be purged if it is empty.

(2) It is not checked whether the Vista translation table is accessible.

(3) UDF for DL/1 can only be purged if no DDM for the file exists.

Administering Different Types of External Objects

The following sections describe all functions for the administration of external objects.

External Objects connected to Predict objects of the following types can be administrated:

- Databases
- Dataspaces
- Files
- Programs
- Storagespaces

Databases

Implemented databases, Predict database objects and generation logs can be processed with functions of the Administration Implemented Database menu. This menu is called with function code L and object code DA in a Predict main menu or with the command ADMINISTRATE DATABASE.

```

13:32:23          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Administration Implemented Database -          Profile HNO

          Function

          C Disconnect implementation
          D Display implementation
          P Purge implementation
          S Select implementation
          U Purge Vista elements

Function .....
Database ID .....
Belongs to VM .....
Database number ....
External object ....*
Only modified ..... N (Y/N)

Restrictions ..*   Profile HNO ,used

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkJl Flip Print Impl AdmFi SelFi Prof Main

```

Parameters	
Database ID	ID of Predict database object connected to the external database. ID must be unique for functions Disconnect implementation, Display implementation and Purge implementation. If asterisk notation is used for function Select implementation, the scope can be limited by the parameters below.
Belongs to VM	Restricts the selection to databases in the specified virtual machine.
Database number	Restricts the selection to databases with the specified physical database number.
External object	Enter the type of external object to be processed. AD Adabas database AT Vista table D2 DB2 database ND Natural DBD blank all
Only modified	Y Restricts the selection to databases that have been modified since generation. N All databases are to be selected, whether they have been modified since generation or not.
Restrictions	See Restrictions in the section Predict User Interface in the Introduction to Predict documentation .

Functions

Display implementation - Code D

For DB2 databases: displays documentation data, generation options and the generation log for the specified database.

For Adabas and IMS databases: only the implementation pointer is displayed.

Purge implementation - Code P

For DB2 databases: Deletes the database and all dependent tablespaces, tables and views. The database ID must be entered again to confirm deletion. If a table holds the last reference to a distinct type, the distinct type is also deleted.

The function can only be applied to one DB2 database at a time. The function is not available for Adabas and IMS databases.

Warning:

Database and all dependent tablespaces, tables and views are **physically deleted** in DB2.

Recovery from this action is not possible.

Disconnect implementation - Code C

Disconnects implemented databases connected to the Predict database object by deleting the implementation pointer of the documentation object.

Select implementation - Code S

This function is used to select a database for further processing with one of the functions above. The following information is given in a single-line list:

- database ID
- database type
- whether the database has been modified since generation
- implementation pointer depending on database type:

Adabas	Adabas database ID, DBnr.
DB2	DB2 database ID, member, library
IMS	NDB name.

Purge Vista elements - Code U

Deletes entries in the Vista translation table.

Enter unique database ID and external object code AT. Other parameters are ignored. A screen appears with all the Vista elements for the specified file. Delete one or more elements by entering P or X in the Cmd column.

If several elements are purged in one transaction, each purge must be confirmed individually.

Dataspaces

DB2 tablespaces/SQL/DS DBspaces, Predict Dataspace objects and generation logs can be processed with functions of the Administration Implemented Dataspace menu. This menu is invoked with code L and object code DC in a Predict main menu or with the command ADMINISTRATE DATASPACE.

```

13:29:43          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Administration Implemented Dataspace -          Profile HNO

          Function

          C Disconnect implementation
          D Display implementation
          P Purge implementation
          S Select implementation

Function .....

Dataspace ID .....
Located in DA .....
Member .....
Library .....
Only modified ..... N (Y/N)

Restrictions .....* Profile HNO ,used

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
    
```

Parameters	
Dataspace ID	ID of Predict Dataspace object connected to the DB2 tablespace or SQL/DS DBspace. ID must be unique for functions Disconnect, Display and Purge implementation.
Contained in DA	For Select implementation function: limits the scope to Dataspaces contained in the specified Predict database object.
Member, Library	If Select function has been executed, these values are displayed, but are not evaluated by any other function.
Restrictions	See Restrictions in the section Predict User Interface in the Introduction to Predict documentation .
Only modified	<p>Y Restricts the selection to dataspace that have been modified since generation.</p> <p>N All dataspace are to be selected, whether they have been modified since generation or not.</p>

Functions

Display implementation - Code D

Displays documentation data, generation options and the generation log for the specified Dataspace.

Purge implementation - Code P

Deletes the tablespace and all dependent tables and views. The dataspace ID must be entered again to confirm deletion. If a table holds the last reference to a distinct type, the distinct type is also deleted.

If dependent tables and views exist, they are listed. The deletion of these dependent objects has to be confirmed with Y.

The function can be applied to one DB2 tablespace at a time.

Warning:

Tablespace and dependent tables and views are **physically deleted** in DB2.

Recovery from this action is not possible.

Disconnect implementation - Code C

Disconnects tablespaces and Predict dataspace by deleting the implementation pointer of the Predict dataspace object.

Select implementation - Code S

This function is used to select a dataspace for further processing with one of the functions above.

The following information is given in a single-line list:

- dataspace ID
- name of tablespace generated from the dataspace
- physical database containing the dataspace
- library and member of implemented dataspace.

Files

Implemented files, Predict file objects and code generated from Predict file objects can be processed with functions of the Administration Implemented File menu. This menu is invoked with code L and object code FI in a Predict main menu or with the command ADMINISTRATION FILE.

```

13:39:46          ***** P R E D I C T  4.3.1  *****          2003-05-31
Plan    0          - Administration Implemented File -          Profile HNO

      Function                                Function

      C Disconnect implementation             P Purge implementation
      D Display implementation               R Refresh file
      N Rename implementation                 S Select implementation
                                           U Purge Vista elements

Function .....

File ID .....                               Files of type ....*
Contained in DA ....                         File number .....
External name .....
Member .....                               Phys Fnr .....
Library .....                               Phys DBnr .....
Only modified ..... N (Y/N)                External object ..*

Restrictions .....*   Profile HNO,used

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelfI Prof Main

```

Parameters	
File ID	ID of Predict file object from which external objects were generated. File ID and external object type must be unique for functions Disconnect, Display and Purge implementation. If asterisk notation is used for function Select, the scope can be limited by the parameters in database, External name, Files of type, and File number.
Contained in DA	Database containing the Predict file object via association <i>Contains FI</i> .
External name	Up to 50 characters can be entered here. Asterisk notation is possible and must be used if external name is longer than 50 characters. This field can be used to denote the full search path of a UNIX file, for example, or to identify an external object without its creator.
Member, Library	For the functions Rename and Purge implementation: if File ID is not sufficient to identify an external object uniquely, Member and Library must be specified. For the functions Disconnect and Display implementation, Member and Library can be specified. For Select implementation, Member and Library can be used to restrict the result list of implementations.
Only modified	Only for function Select. Y Restricts the function to external objects where the corresponding file object has been modified since generation. N All external objects are to be included, whether the corresponding file object has been modified since generation/incorporation or not.
Files of type	Not all file types are valid for all functions. Enter an asterisk in this field to display the valid values for the selected function.
File number	Logical file number.
Phys Fnr Phys DBnr	Physical file number/database number of Adabas file. If these parameters are used, Member and Library are ignored.
External object	Type of external object (language). Must be entered for all functions except Select implementation to identify the object uniquely. With the function Select, this field may be left blank to select files of all external object types.
Restrictions	See Restrictions in the section Predict User Interface in the Introduction to Predict documentation .

Functions

Disconnect implementation - Code C

Disconnects implemented external object(s) from the Predict file object by deleting the implementation pointer of the file object.

Member and library must be specified (for Adabas files the physical database/file number).

With files of type AT, all files used for generation are disconnected in a single operation.

Display implementation - Code D

Displays documentation data, generation options and generated code of specified file ID and external object type.

Member and library may be specified to identify an external object uniquely.

If this information is not entered and more than one external object is found, each object is displayed in sequence.

Rename implementation - Code N

Moves generated code to another member and/or to another library. Specify File ID, External object type Library and Member of the code to be moved. The new member and/or the new library are entered in a second screen.

DDMs, Super Natural files and processing rules cannot be renamed with this function. To rename objects of these types, they have to be generated again.

Purge implementation - Code P

Deletes generated code. File ID must be identified uniquely, either by library and member or, for Adabas files, by physical file number/database number. If a DB2 table holds the last reference to a distinct type, the distinct type is also deleted.

Warning:

File is physically deleted in Adabas, DB2 or Adabas SQL Server directory. All dependent objects are deleted, too.

File ID and External object must be specified when deleting all types of external objects. In addition, Library and Member must be specified when deleting generated code.

Note:

Special rules apply when deleting Adabas files or DDMs:

Adabas files

The following rules apply:

- Parameters DB-Nr and File-Nr must be specified.
- If the file to be deleted is in use, the Adabas option Stop user using file can be used.
- If Predict/AOS Security is active, access rights are required to delete an Adabas file or to stop users using the file.

See Protecting Adabas Databases and Files in the section **Protecting External Objects in Predict with Natural Security** in the **Predict Security documentation**.

DDMs

- For DDMs defined in Natural Security, access rights to maintain the security definition of the file in Natural Security are required.
See Protecting DDMs in the section **Protecting External Objects in Predict with Natural Security** in the **Predict Security documentation**.

Refresh file - Code R

Deletes all records stored in Adabas files or DB2 tables. Data structure remains intact. Refresh operation must be confirmed.

Special rules apply when refreshing Adabas files:

- DB-Nr and File-Nr must be specified.
- If the file to be refreshed is in use, the Adabas option Stop user using file can be used.

Note:

If Predict/AOS Security is active, access rights are required to delete an Adabas file or to stop users using the file. See Protecting Adabas Databases and Files in the section **Protecting External Objects in Predict with Natural Security** in the **Predict Security documentation**.

Warning:

This function deletes the entire contents of an Adabas file or DB2 table/view. Recovery from this action is not possible.

Select implementation - Code S

Displays a list of existing generated code sorted by file ID of the Predict file objects that were used to generate the code. Asterisk notation is possible for File ID.

Scope can be limited by parameters described above and by additional Restrictions.

The following information is given in a single-line list:

- ID of Predict file object from which the code or object was generated
- external object type
- whether the file object has been modified after generation
- for implemented Adabas files: database ID and file number
- for 3GL preprocessor data: library containing XRef data, for example *SYSCOB* for COBOL copy code
- for other objects: the name of the member and library where the implemented code is stored.

Note:

If this function is used for external object type AT, all Vista elements are listed, without checking whether they can be accessed by the user or not.

Note:

If no external object type is specified, and for the same file both Adabas files and Vista elements exist, only the Vista tables are listed with this function. If Vista elements exist, an Adabas file must exist for the specified file.

The parameter Only modified refers to the Adabas file.

Note:

When you specify a database, all the implementations pointing to that database are listed, but if objects which have further implementations (for example in other databases) are found, all these links are displayed as well.

Purge Vista elements - Code U

Deletes entries in the Vista translation table.

Enter unique file ID and external object code AT. Other parameters are ignored. A screen appears with all the Vista elements for the specified file. Delete one or more elements by entering P or X in the Cmd column.

If several elements are purged in one transaction, each purge must be confirmed individually.

Programs

DB2 procedures, DB2 database functions and generation logs can be processed with functions of the Administration Implemented Program menu. This menu is invoked with code L and object code PR in a Predict main menu or with the command ADMINISTRATE PROGRAM.

```
13:39:05          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan 0            Administration Implemented Program          Profile HNO

Function

C Disconnect implementation
D Display implementation
P Purge implementation
S Select implementation

Function .....

Program ID .....          Program of type ..*
Member .....          Language .....*
Library .....
Only modified .... N (N/Y)
Belongs to SY ....
Restrictions ....* Profile HNO ,used

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main
```

Parameters	
Program ID	ID of Predict program object from which external objects were generated. ID must be unique for functions Disconnect, Display and Purge implementation. If asterisk notation is used for function Select, the scope can be limited by the parameters below.
Program of type	R Only programs of type SQL procedure. U Only programs of type Database function. blank All programs.
Language	Only available for programs of type R. Valid values: B BAL/Assembler C COBOL H Language C N Natural P PL/I R REXX S SQL procedure lang. V Java
Member, Library	If Select function has been executed, these values are displayed, but are not evaluated by any other function.
Only modified	Y Restricts the selection to programs that have been modified since generation. N All programs are to be selected, whether they have been modified since generation or not.
Restrictions	See Restrictions in the section Predict User Interface in the Introduction to Predict documentation .

Functions

Disconnect implementation - Code C

Disconnects implemented external object(s) from the Predict program specified. The implementation pointer from the documentation object to the external object(s) is deleted, but the objects themselves remain intact.

Display implementation - Code D

Displays documentation data, generation options and generated code of the specified program.

Purge implementation - Code P

Deletes the procedure/function in DB2.

Select implementation - Code S

Displays a list of existing generated code sorted by program ID of the Predict entries that were used to generate the code. Asterisk notation is possible for program ID.

Scope can also be limited by parameters Member and Library and by additional Restrictions. The following information is given in a single-line list:

- Program ID,
- name of the procedure/function generated from the Predict program object,
- library and member of the implemented procedure/function.

Storagespaces

DB2 storagegroups, Predict storagespace objects and generation logs can be processed with functions of the Administration Implemented Storagespace menu. This menu is invoked with code L and object code ST in a Predict main menu or with the command ADMINISTRATE STORAGESPACE.

```

13:39:05          ***** P R E D I C T 4.3.1 *****          2003-05-31
Plan    0          Administration Implemented Storagespace      Profile HNO

          Function

          C Disconnect implementation
          D Display implementation
          P Purge implementation
          S Select implementation

Function .....

Storagespace ID ..
Member .....
Library .....
Only modified .... N (N/Y)

Restrictions ....* Profile HNO ,used

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnkEl Flip Print Impl AdmFi SelFi Prof Main

```

Parameters	
Storagespace ID	ID of Predict storagespace object from which external objects were generated. ID must be unique for functions Disconnect, Display and Purge implementation. If asterisk notation is used for function Select, the scope can be limited by the parameters below.
Member, Library	If Select function has been executed, these values are displayed, but are not evaluated by any other function.
Only modified	<p>Y Restricts the selection to storagespaces that have been modified since generation.</p> <p>N All storagespaces are to be selected, whether they have been modified since generation or not.</p>
Restrictions	See Restrictions in the section Predict User Interface in the Introduction to Predict documentation .

Functions

Disconnect implementation - Code C

Disconnects implemented external object(s) from the Predict storagespace by deleting the implementation pointer of the documentation object.

Display implementation - Code D

Displays documentation data, generation options and generated code of specified storagespace.

Purge implementation - Code P

Deletes generated code.

A storagespace which is used by any tablespace or index space cannot be deleted. In this case, a list of all the tablespaces, tables and views used by the storagespace is displayed.

Warning:

Storagegroup is physically deleted in DB2.

Select implementation - Code S

Displays list of existing generated code sorted by storagespace ID of the Predict file objects that were used to generate the code. Asterisk notation is possible for storagespace ID.

Scope can also be limited by parameters Member and Library and by additional Restrictions.

The following information is given in a single-line list:

- Storagespace ID
- name of storagegroup generated from the Predict storagespace object
- library and member of implemented storagegroup.

Preprocessor

This section covers the following topics:

- Using the Preprocessor
 - Preprocessor Statements
 - Creation of XRef Data
-

Overview

The Predict preprocessor offers two main options:

- XRef data on the use of copy/include code and on the call of external members in 3GL programs is written. For Assembler, preprocessor statements have to be included into source code to perform this operation. This option is completely independent from Predict documentation objects. See Creation of XRef Data for a description.
- Data definitions generated from Predict file objects can be included into COBOL, PL/I and Assembler (BAL, 370-Assembler) programs and XRef data documenting the use of these definitions is written.

Data definitions generated from Predict objects can be included in two ways:

- the Preprocessor copies file layouts into a source program that have already been generated in Predict as copy code.
- the Preprocessor generates file layouts from Predict file objects and copies these file layouts into a source program.

Two types of definitions can be included:

- record buffer structures
- format buffer structures

Using the Preprocessor

Using the preprocessor is a two-step process:

1. Write preprocessor statements to the source code of programs to be processed. See also Preprocessor Statements.
2. Call the preprocessor for the programs to be processed. See Calling the Preprocessor below.

The preprocessor

- scans the source program for preprocessor statements and for certain other statements. See the section Creation of XRef Data.
- generates copy code from Predict file objects and inserts this code into the source program (or inserts already generated code into the source program)
- writes corresponding XRef data.

Using the Preprocessor with Adabas Native SQL

To use the preprocessor in combination with Adabas Native SQL, it must be called before Adabas Native SQL. An asterisk must then be specified as Member name.

Calling the Preprocessor

The preprocessor is called with the following command:

```
PREPROCESS <language> [<member name>] [<library>] [<user-ID>]
```

If the generation default parameter Preprocessor force is set to Y, the member must be documented in Predict as a 3GL program with fully qualified implementation pointer (member name and library name).

Parameters	
language	Language in which the program to be processed is written: <ul style="list-style-type: none"> ● COBOL ● PL/I ● BAL Identifies 370-Assembler.
member name	The member name must not be longer than eight characters. The member name can be supplied in three ways. If the member name is specified in more than one way, it is evaluated in the order below: <ul style="list-style-type: none"> ● When calling the Preprocessor; ● Using the preprocessor statement PROGRAM; ● Using a normal source program statement <ul style="list-style-type: none"> ○ in COBOL, the PROGRAM-ID in the Identification division; ○ in PL/I, the label preceding the first PROC or PROCEDURE statement ○ in BAL, the symbol in the name field of the START statement or the first CSECT statement. If the preprocessor is used in combination with Adabas Native SQL, an asterisk must be specified.
library	If the parameter library is specified, a system of type G (3GL application) containing this library name in its implementation pointer must already have been defined. If no library is specified, the default libraries *SYSCOB*, *SYSBAL* or *SYSPLI* are used.
user-ID	The parameter user ID is only evaluated in batch mode. Online, the session user ID is evaluated.

Prerequisites - Work Files

The PREPROCESS command requires the first three Natural workfiles (DD-names CMWKF01, CMWKF02 and CMWKF03; BS2000-link-names W01, W02 and W03).

- The first workfile is the input data set for the preprocessor.
- The second workfile is the output data set; it should be declared with a record length of 80.
- The third workfile is a temporary workfile; it should have record format F (fixed) or FB (fixed blocked) and the record size must be 91.

Note:

For an explanation of the Preprocessor Condition Codes see the section Predict in Batch Mode in the **Predict Administration documentation**.

Reports - Print Files

The preprocessor produces three types of reports:

- a report that the run was successful or unsuccessful
- a list of preprocessor commands found
- a list of errors.

In batch, the reports are written to the first three Natural reports (DD-names CMPRT01, CMPRT02 and CMPRT03; BS2000-link-names P01, P02 and P03).

Preprocessor Statements

Overview of Preprocessor Statements

Statements to Include Copy Code Generated from Predict file objects

The preprocessor statements COPY, FORMAT-BUFFER and GENERATE in COBOL, PL/I and Assembler programs instruct the preprocessor to include data definitions or a format buffer.

COPY

Instructs the preprocessor to insert copy code which has previously been generated by Predict. XRef data is written for the file and each field in the file.

FORMAT-BUFFER

Instructs the preprocessor to generate an Adabas format buffer and insert it at the position of the statement. XRef data is written for the file and each field in the file.

GENERATE

Instructs the preprocessor to generate a record buffer and optionally a format buffer and insert it at the position of the statement. XRef data is written for the file and each field in the file.

Statements to Write XRef Data for 3GL Copy/Include Code or Function Calls

The preprocessor statements ENTRY and CALL - if included in Assembler programs - instruct the preprocessor to write XRef data for entry points or the call of external programs. Information to be written to XRef data is specified in parameters of the statement.

CALL

Specifies the name of a called external program or function that is to be stored in the active reference records. This command can only be used for Assembler programs.

ENTRY

Specifies the name of a program entry that is to be stored in the active reference records. This command can only be used for Assembler programs.

PROGRAM

Member ID used in XRef data. This statement is interpreted only if no member name is specified when the Preprocessor is called.

Format of Preprocessor Statements

The following rules apply to preprocessor statements:

- Preprocessor statements start with EXEC ADABAS (there may be any number of blanks between the two words, but both must be coded on the same line).
- Preprocessor statements can be terminated in any of the following ways:
 - with the statement END-EXEC
 - in COBOL by a period (.)
 - in PL/I by a semicolon (;)
 - In BAL, the preprocessor terminates processing at the end of the current card unless a continuation character is punched in column 72.
- The preprocessor assumes standard statement layout in BAL, for example the Assembler statement ICTL is

not valid.

- Preprocessor control statements are left in the source program as comments.

Using Keyword and/or Positional Parameters

- Parameters of preprocessor statements can be specified in positional or keyword form. Both forms are described below.
- Keyword and positional parameters can be mixed. This allows the use of a keyword parameter as a starting point for subsequent positional parameters, as shown in the following example:
For Assembler, the INIT parameter is the eighth parameter of the EXEC Adabas GENERATE statement. VALIDATION is the next parameter in the list and can therefore be specified as a positional parameter directly behind the INIT parameter.

```
EXEC ADABAS GENERATE <file-name>,INIT=<init>,<validation>
```

CALL

The name of a called external program or function can be specified for Assembler programs. The name is stored in the active reference records.

Syntax with Positional Parameters

```
EXEC ADABAS CALL <function-name>
                END-EXEC
```

Syntax with Keyword Parameters

```
EXEC ADABAS CALL FUNCTION=<function-name>
                END-EXEC
```

Parameters	
function-name	The name of a called external program or function that should be stored in the active reference records. Maximum length: 8 characters.

COPY

Instructs the preprocessor to include copy code previously generated by Predict. If more than one copy code member has been generated for the appropriate file and language, the copy code name must be specified. XRef data is written for the file and each field in the file.

Copying a Member from a File that has been Modified after Generation

This statement can also be used if the file has been modified after the corresponding member was generated. The system behavior depends on the Predict version with which the member was generated and on parameter Ignore changes. See table below.

Syntax

with Positional Parameters

```
EXEC ADABAS COPY <file-name><copycode-name> <ignore-changes>
                END-EXEC
```

with Keyword Parameters

```
EXEC ADABAS COPY FILE=<file-name>
      MEM=<copycode-name>
      IGNORE-CHANGES=<ignore-changes>
      END-EXEC
```

Note:

All parameters except <file-name> are optional.

Parameters	
file name	The ID of the Predict file object from which the copy code was generated. See GENERATE for a list of file types that can be used for generating copy code.
member name	The member name which was supplied during copy code generation. Maximum length: 8 characters.
ignore changes	<p>This parameter determines whether a member connected to a file that has been modified after generation is copied.</p> <p>N Default. A member connected to a file that has been modified after generation is not copied. The Preprocessor issues an error message and terminates with condition code 107.</p> <p>Y A member connected to a modified file is copied - irrespective of whether the member contains usage information. With this option, the preprocessor writes the XRef data on the basis of the current documentation of the file in Predict. A warning is given if the file object has been modified after generation, since using this option may result in inconsistent XRef data. See Additional Usage Information for Fields in a File.</p> <p>X A member of a modified file is only copied if it was generated with Predict version 3.3 or above. With this version, additional information on the usage of fields in the file is available. This information is used by the Preprocessor when writing XRef data. See Additional Usage Information for Fields in a File. If the file has been modified since generation and the usage information is not available because the member was generated with an earlier version of Predict, the Preprocessor issues an error message and terminates with condition code 107.</p>

ENTRY

Declares the name of an entry point in Assembler programs. The name is stored in the active reference records.

Syntax with Positional Parameters

```
EXEC ADABAS ENTRY <entry-name>
      END-EXEC
```

Syntax with Keyword Parameters

```
EXEC ADABAS ENTRY FUNCTION=<entry-name>
      END-EXEC
```

Parameters	
entry-name	The name of an entry point to be stored in the XRef data. Maximum length: 8 characters.

FORMAT-BUFFER

The FORMAT-BUFFER statement instructs the preprocessor to generate an Adabas format buffer and insert it into the 3GL member at the position of the statement. XRef data is written for the file and each field in the file.

Syntax with Positional Parameters

```
EXEC ADABAS FORMAT-BUFFER
    <file-name><format-buffer-name>
    <sync><offset><dcl>
    <adabas-version><buffer-format>
    <literal-delimiter>
END-EXEC
```

Syntax with Keyword Parameters

```
EXEC ADABAS FORMAT-BUFFER
    FILE=<file-name>
    FORMAT-BUFFER-NAME=<format-buffer-name>
    SYNC=<sync>
    OFFSET=<offset>
    DCL=<dcl>
    ADA-VER=<adabas-version>
    FORMAT=<buffer-format>
    DELIMITER=<literal-delimiter>
END-EXEC
```

Note:

All parameters except <file-name> are optional.

Parameters	
file-name	Specifies the ID of the file object in Predict. This parameter is mandatory. The file can be one of the following types: <ul style="list-style-type: none"> ● Adabas file (type A), where parameter Adabas SQL usage may not be set to Y. ● Adabas userview (type U)
format-buffer-name	The name to be given to the format buffer in the target program. In COBOL, this may be up to 30 characters long; in PL/I, up to 31; and in BAL, up to 8 characters. The default value is the same as in the respective generation function.
sync	Y All appropriate fields will be aligned. N No fields will be aligned. S Fields will be aligned only if the corresponding Predict object has the 3GL specification Synchronized=S.
offset	L,Y,P The total length of the code will be included in the format buffer. V A constant will be generated for the format buffer length.
dcl	Only applies to PL/I include code. Y The generated code will be preceded by a declare statement DCL and will end with a semicolon instead of a comma.
adabas-version	The version of Adabas for which the copy code of the Adabas files and userviews is to be generated. See list of possible values in the section Adabas Version.
buffer-format	Y Normal format for format buffer. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible. F Full format for format buffer. The format buffer will include field length and format.
literal-delimiter	Only valid for COBOL copy code. S single quotes D double quotes

GENERATE

The statement GENERATE instructs the preprocessor to generate copy code from Predict file objects and insert it into the 3GL member at the position of the statement. XRef data is written for the file and each field in the file.

Generation can be performed for Assembler, COBOL and PL/I. See table of valid file types and languages below.

The generated copy code will be written to the source area and then copied into the source program.

Code	File Type	Assembler	COBOL	PL/I
A	Adabas file	Y	Y	Y
B	Adabas SQL view	Y	Y	Y
BT, BV	Adabas D table/view		Y	
D, E	DB2 table/view	Y	Y	Y
F	rdb file	Y	Y	Y
J	IMS segment layout	Y	Y	Y
JT, JV	Ingres table/view		Y	Y
L	logical VSAM file	Y	Y	Y
M	ISAM file	Y	Y	Y
O	other file	Y	Y	Y
OT, OV	Oracle table/view		Y	Y
S	sequential file	Y	Y	Y
T	rms file	Y	Y	Y
U	Adabas user view	Y	Y	Y
V	VSAM file	Y	Y	Y
X	General SQL file	Y	Y	Y
XT, XV	Informix table/view		Y	
YT, YV	Sybase table/view		Y	
1	LEASY	Y	Y	Y
2	ISAM BS2000	Y	Y	Y

Generating Assembler Copy Code

Syntax with Positional Parameters

```
EXEC ADABAS GENERATE <filename><prefix><suffix><dsect>
                    <dc-ds><nr-comments><offset><init>
                    <validation><truncation><dsect-name><align>
                    <equ><adabas-version><generate-format-buffer>
                    <format-buffer-name><counter-length>
END-EXEC
```

Syntax with Keyword Parameters

```
EXEC ADABAS GENERATE FILE=<file-name>
                    PREFIX=<prefix>
                    SUFFIX=<suffix>
                    DSECT=<dsect>
                    DC-DS=<dc-ds>
                    NR-COMMENTS=<nr-comments>
```

```

OFFSET=<offset>
INIT=<init>
VALIDATION=<validation>
TRUNCATION=<truncation>
RECORD-BUFFER-NAME=<dsect-name>
SYNC=<align>
EQU=<equ>
ADA-VER=<adabas-version>
FORMAT-BUFFER=<generate-format-buffer>
FORMAT-BUFFER-NAME=<format-buffer-name>
COUNTER-LENGTH=<counter-length>
END-EXEC
    
```

Note:

All parameters except <file-name> are optional.

Details of the parameters are given in the following table and also in the section Generation in this documentation. If a parameter is omitted, the default value defined in Predict by the DDA will be used. See the section Defaults in the **Predict Administration documentation**.

If the statement is entered using positional parameters and no prefix or no suffix is required, an asterisk (*) should be substituted for the parameter <prefix> or <suffix>.

Parameters	
file-name	ID of the Predict file object from which the Assembler copy code is to be generated. This parameter is mandatory. See table of valid file types in the description of Parameter Generate.
prefix	A prefix for the field names. Maximum length: 8 characters.
suffix	A suffix for the field names. Maximum length: 8 characters.
dsect	Y The copy code will be generated as an ASSEMBLER DSECT (dummy section). The DSECT will have the name specified by the <dsect-name> parameter, or the file ID if no <dsect-name> parameter is supplied.
dc-ds	DC Assembler DC (define constant) instructions will be generated. DS Assembler DS (define storage) instructions will be generated.
nr-comments	Specifies the number of abstract lines per field (0-16) which will be included in the generated code.

offset	<p>Y Include the offset of each item in the record buffer structure (relative to the beginning of the structure) in decimal and hexadecimal formats as a comment. The total length of each buffer is also included.</p> <p>P Include the absolute position (offset+1) as a comment.</p> <p>L Include the total lengths of the record buffer and the format buffer as a comment.</p> <p>V Only allowed if parameters As DSECT=N and With DC or DS=DC. The file number and the calculated lengths of the record buffer and the format buffer are to be generated as constants in the copy code. The name of the file number constant is the record buffer name with N as prefix. The name of each length constant is the appropriate buffer name with L as prefix. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.</p> <p>N No offset.</p>
init	<p>This option takes effect only when With DC or DS=DC and As DSECT is set to N.</p> <p>N No initialization.</p> <p>Y Statements are generated to initialize the structure with the value specified for Init value in the corresponding field object in Predict. Fields with no value for Init value are initialized with zeros or blanks. Aligned 8-digit fields with format B or I are not initialized. In a PE group with the 3GL specification Gr.structur set to blank, only the first occurrence of each field is initialized.</p>
validation	<p>Determines how invalid characters in a field name are handled.</p> <p>blank Invalid characters will result in an error message but will not be deleted.</p> <p>rep.char. Invalid characters will be replaced by this character. Valid values: letters A-Z, digits 0-9, \$, % or #.</p> <p>* Invalid characters will be deleted.</p>
truncation	<p>Specifies which characters are deleted if a generated field name is longer than 8 characters:</p> <p>L truncate from the left</p> <p>R truncate from the right</p> <p>M truncate from the middle.</p>

dsect-name	Specifies the name of the record buffer in the generated structure. The effect of this parameter depends on parameter dsect.
align	<p>Y All appropriate fields will be aligned.</p> <p>N No fields will be aligned.</p> <p>S Fields will be aligned only if the corresponding Predict object has the 3GL specification Synchronized= S.</p>
equ	<p>Y EQU statements are to be generated for fields of length 1 whose format is not P and comment lines are to be generated for other fields, using any condition names defined as attributes of the Predict field objects. These names are prefixed, suffixed, validated and truncated in the same way as field names.</p> <p>For fields of format L where no condition name was specified, a condition name is generated by concatenating the field name "Example" to the prefix N.</p> <p>In this case the following statement is generated:</p> <p>NEXAMPLE EQU X'00'.</p>
adabas-version	The version of Adabas for which the copy code of the Adabas files and userviews is to be generated. See list of possible values in the section Adabas Version.
generate-format-buffer	<p>Format buffer generation for Assembler copy code is only allowed if parameters As DSECT=N and With DC or DS=DC.</p> <p>The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas SQL usage set to N) or for files of type U.</p> <p>Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, set this parameter to F or N.</p>
format-buffer-name	Specifies the label (name) of the format buffer in the generated structure. If omitted, the file ID prefixed by F is used.
counter-length	Length of additional counter fields: Valid values: 1, 2.

Generating COBOL Copy Code

Syntax with Positional Parameters

```
EXEC ADABAS GENERATE <file-name><prefix><suffix> <start-level>
                    <level-increment><shift-number><nr-comments>
                    <offset><init><validation><truncation>
                    <record-buffer-name><cond-name><sync>
                    <indexed><depending>
                    <adabas-version><generate-format-buffer>
                    <format-buffer-name>
                    <check-name>
                    <literal-delimiter>
                    <decimal-char>
                    <redefine-name>
END-EXEC
```

Syntax with Keyword Parameters

```
EXEC ADABAS GENERATE FILE=<file-name>
                    PREFIX=<prefix>
                    SUFFIX=<suffix>
                    START-LEVEL=<start-level>
                    LEVEL-INCREMENT=<level-increment>
                    SHIFT-NUMBER=<shift-number>
                    NR-COMMENTS=<nr-comments>
                    OFFSET=<offset>
                    INIT=<init>
                    VALIDATION=<validation>
                    TRUNCATION=<truncation>
                    RECORD-BUFFER-NAME=<record-buffer-name>
                    COND-NAME=<cond-name>
                    SYNC=<sync>
                    INDEXED=<indexed>
                    DEPENDING=<depending>
                    ADA-VER=<adabas-version>
                    FORMAT-BUFFER=<generate-format-buffer>
                    FORMAT-BUFFER-NAME=<format-buffer-name>
                    CHECK-NAME=<check-name>
                    DELIMITER=<literal-delimiter>
                    DEC-CHAR=<decimal-char>
                    REDEFINE-NAME=<redefine-name>
END-EXEC
```

Note:

All parameters except <file-name> are optional.

Details of the parameters are given in the following table and also in the section Generation in this documentation. If a parameter is omitted, the default value defined in Predict by the DDA will be used. See the section Defaults in the **Predict Administration documentation**.

If the statement is entered using positional parameters and no prefix or no suffix is required, an asterisk (*) should be substituted for the parameter <prefix> or <suffix>.

Parameters	
file-name	ID of the Predict file object from which the COBOL copy code is to be generated. This parameter is mandatory. See table of valid file types in the description of Parameter Generate.

prefix	A prefix for the field names. Maximum length: 16 characters.
suffix	A suffix for the field names. Maximum length: 16 characters.
start-level	Specifies the starting level of the generated record buffer. Valid values are in the range 1 - 40.
level-increment	Specifies the level-increment. Valid values are in the range 1 - 40.
shift-number	The number of positions to be shifted right when a level number which is higher than the current level number is encountered. Valid values are in the range 0 - 9.
nr-comments	Specifies the number of abstract lines per field that will be included in the generated code. Valid values are in the range 0 - 16.
offset	<p>Y The offset of each item in the record buffer structure (relative to the beginning of the structure) in decimal and hexadecimal formats is to be included as a comment. The total length of each buffer is also included.</p> <p>P As above, but the absolute position (offset+1) is included as a comment.</p> <p>L The total lengths of the record buffer and the format buffer are to be included as a comment.</p> <p>V The file number and the calculated lengths of the record buffer and the format buffer are to be generated as constants in the copy code. The name of the file number constant is the record buffer name prefixed by N-. The name of each length constant is the appropriate buffer name prefixed by L-. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.</p> <p>N No offset.</p>
init	<p>Y The fields will be initialized wherever possible using a COBOL VALUE clause. Any fields with INIT VALUES in their Predict objects will be initialized with those values; other fields will be initialized with low values (zeros or spaces).</p> <p>S Only fields with INIT VALUES in the corresponding Predict object will be initialized.</p> <p>N No initialization.</p>

validation	<p>Determines how invalid characters in a field name are handled.</p> <p>blank Invalid characters will result in an error message but will not be deleted.</p> <p>rep.char. Invalid characters will be replaced by this character. Valid values: letters A-Z, digits 0-9 or hyphen.</p> <p>*</p> <p>Invalid characters will be deleted.</p>
truncation	<p>Specifies which characters are deleted if a generated field name is longer than 30 characters:</p> <p>L truncate from the left</p> <p>R truncate from the right</p> <p>M truncate from the middle.</p>
record-buffer-name	<p>Specifies the name of the record buffer in the generated structure. If omitted, the file ID is used.</p>
cond-name	<p>Y Any condition names defined in the Predict field objects are to be generated on level 88, provided that the respective field objects have one of the following formats:</p> <ul style="list-style-type: none"> ● A All lengths ● N or P Less than 19 digits ● I or B 2, 4 or 8 digits ● L A FALSE-condition will always be generated. The Condition name is then generated by concatenating the field name to the prefix N- (if not specified explicitly). <p>These names are prefixed, suffixed, validated and truncated in the same way as field names.</p>
sync	<p>Y All appropriate fields will be aligned.</p> <p>N No fields will be aligned.</p> <p>S Fields will be aligned only if the corresponding Predict object has the 3GL specification Synchronized = S.</p>

indexed	<p>Y The COBOL clause INDEXED BY will be generated for all repetitive fields (MU/MC and PE/PC).</p> <p>S This clause will be generated only for repetitive fields which have INDEXED BY NAMES in their Predict objects.</p> <p>N This clause will not be generated for any field.</p>
depending	<p>Y COBOL attribute OCCURS DEPENDING ON is generated for a field or field group in a file if it has type PE or MU and DEPENDING ON NAME is specified for this field. These names are prefixed, suffixed, validated and truncated in the same way as field names.</p> <p>Note: This option is not allowed for files of type A or U. This option is ignored when using a WANG COBOL compiler.</p>
adabas-version	<p>The version of Adabas for which the copy code of the Adabas files and userviews is to be generated. See table of valid values in the section Adabas Version.</p>
generate-format-buffer	<p>The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas SQL usage set to N) or for files of type U.</p> <p>Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, set this parameter to F or N.</p>
format-buffer-name	<p>Specifies the name of the format buffer in the generated structure. If omitted, the file ID prefixed by FORMAT-BUFFER- is used.</p>
check-name	<p>A COBOL field names are checked for uniqueness throughout the whole structure.</p> <p>Y Structure levels are included in the validation check of the field names: if two fields have the same name, they must be separated by at least one field with a different name and a lower-level number.</p> <p>N No check for duplicate field names is performed</p>

literal-delimiter	<p>S single quotes,</p> <p>D double quotes</p>
decimal-character	<p>P point,</p> <p>C comma</p>
redefine-name	<p>Determines how COBOL field names for Predict fields of type RE are generated:</p> <p>F The string FILLER is used as redefinition name.</p> <p>S The suffix REGR is added to the Predict field ID. If a field is redefined more than once, the suffix will have the form REGRn, where n is an integer incremented by 1 for each field of type RE.</p>

Generating PL/I Include Code

Syntax with Positional Parameters

```
EXEC ADABAS GENERATE <file-name><prefix><suffix>
                    <start-level><level-increment><shift-number>
                    <nr-comments><offset><init><struct-as-char>
                    <static><validation><truncation>
                    <record-buffer-name><align><dcl>
                    <adabas-version><generate-format-buffer>
                    <format-buffer-name><check-name>
                    <numeric sign><position of sign>
END-EXEC
```

Syntax with Keyword Parameters

```
EXEC ADABAS GENERATE FILE=<file-name>
                    PREFIX=<prefix>
                    SUFFIX=<suffix>
                    START-LEVEL=<start-level>
                    LEVEL-INCREMENT=<level-increment>
                    SHIFT-NUMBER=<shift-number>
                    NR-COMMENTS=<nr-comments>
                    OFFSET=<offset>
                    INIT=<init>
                    STRUCTURE=<struct-as-char>
                    STATIC=<static>
                    VALIDATION=<validation>
                    TRUNCATION=<truncation>
                    RECORD-BUFFER-NAME=<record-buffer-name>
                    SYNC=<align>
                    DCL=<dcl>
                    ADA-VER=<adabas-version>
                    FORMAT-BUFFER=<generate-format-buffer>
                    FORMAT-BUFFER-NAME=<format-buffer-name>
```

```
CHECK-NAME=<check-name>
NUM-SIGN=<numeric sign>
POS-SIGN=<position of sign>
END-EXEC
```

Note:

All parameters except <file-name> are optional.

Details of the parameters are given in the following table and also in the section Generation in this documentation. If a parameter is omitted, the default value defined in Predict by the DDA will be used. See the section Defaults in the **Predict Administration documentation**.

If the statement is entered using positional parameters and no prefix or no suffix is required, an asterisk (*) should be substituted for the parameter <prefix> or <suffix>.

Parameters	
file-name	ID of the Predict file object from which the PL/I include code is to be generated. This parameter is mandatory. See table of valid file types in the description of Parameter Generate.
prefix	Specifies a prefix for the field names. Maximum length: 16 characters.
suffix	Specifies a suffix for the field names. Maximum length: 16 characters.
start-level	Specifies the starting level of the generated record buffer. Valid values are in the range 1 - 40.
level-increment	Specifies the level-increment. Valid values are in the range 1 - 40.
shift-number	The number of positions to be shifted right when a level number which is higher than the current level number is encountered. Valid values are in the range 0 - 9.
nr-comments	Specifies the number of abstract lines per field that will be included in the generated code. Valid values are in the range 0 - 16.

offset	<p>Y The offset of each item in the record buffer structure (relative to the beginning of the structure) in decimal and hexadecimal formats is to be included as a comment. The total length of each buffer is also included.</p> <p>P As above, but the absolute position (offset+1) is included as a comment.</p> <p>L The total lengths of the record buffer and the format buffer are to be included as a comment.</p> <p>V The file number and the calculated lengths of the record buffer and the format buffer are to be generated as constants in the include code. The name of the file number constant will be the record buffer name prefixed by N_. The name of each length constant will be the appropriate buffer name prefixed by L_. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.</p> <p>N No offset.</p>
init	<p>Y The fields will be initialized wherever possible. Any fields with INIT VALUEs in their Predict objects will be initialized with those values; other fields will be initialized with low values (zeros or spaces).</p> <p>S Only fields with INIT VALUEs in the corresponding Predict object will be initialized.</p> <p>N No initialization.</p>
struct-as-char	<p>Y The entire generated structure will be declared at the end of the record buffer as a single character-string.</p>
static	<p>Y The structure will be declared with the attribute <code>STATIC</code>.</p>
validation	<p>Determines how invalid characters in a field name are handled.</p> <p>blank Invalid characters will result in an error message but will not be deleted.</p> <p>rep.char. Invalid characters will be replaced by this character. Valid values: letters A-Z, digits 0-9, \$, %, # or _ (underscore).</p> <p>* Invalid characters will be deleted.</p>

truncation	<p>Specifies which characters are deleted if a generated field name is longer than 31 characters:</p> <p>L truncate from the left</p> <p>R truncate from the right</p> <p>M truncate from the middle.</p>
record-buffer-name	Specifies the name of the record buffer in the generated structure. If omitted, the file ID is used.
align	<p>Y All appropriate fields will be aligned.</p> <p>N No fields will be aligned.</p> <p>S Fields will be aligned only if the corresponding Predict object has the 3GL specification Synchronized= S.</p> <p>Note: This parameter only takes effect with fields that have the PL/I attribute FIXED BIN or FLOAT DEC.</p>
dcl	<p>Y The generated code will be preceded by a declare statement DCL and will end with a semicolon instead of a comma.</p>
adabas-version	The version of Adabas for which the include code of the Adabas files and userviews is to be generated. See table of valid values in the section Adabas Version.
generate-format-buffer	<p>The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas SQL usage set to N) or for files of type U.</p> <p>Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, set this parameter to F or N.</p>
format-buffer-name	Specifies the name of the format buffer in the generated structure. If omitted, the file ID prefixed by FORBUF_ is used.

check-name	<p>A Field names are checked for uniqueness throughout the whole structure.</p> <p>Y Structure levels are included in the validation check of the field names: if two fields have the same name, they must be separated by at least one field with a different name and a lower level number.</p> <p>N No check for duplicate field names is performed</p>
numeric sign	Specifies which of the PL/I picture characters T, I or R is to be used for the representation of numeric values of format packed with sign or unpacked with sign.
position of sign	<p>Defines the position of the sign in a numeric field:</p> <p>L left</p> <p>R right.</p>

PROGRAM

The member name can be passed to the preprocessor with the PROGRAM statement:

Syntax with Positional Parameters

```
EXEC ADABAS PROGRAM <member-name><library>
                    END-EXEC
```

Syntax with Keyword Parameters

```
EXEC ADABAS PROGRAM
                    PROGRAM-ID=<member-name>
                    LIBRARY-ID=<library>
                    END-EXEC
```

Note:

If member and library are specified when the Preprocessor is called, these values are taken. The statement EXEC ADABAS PROGRAM is then not necessary.

Parameters	
member-name	The name used to identify the XRef data. Maximum length: 8 characters.
library	<p>If the parameter library is specified, a system of type G (3GL application) that contains this library name in its implementation pointer must have been defined before.</p> <p>If no library is specified, the *SYSCOB*, *SYSBAL* or *SYSPLI* libraries are used.</p>

Creation of XRef Data

XRef data is written for Assembler, COBOL, or PL/I programs. The following general rules apply when writing XRef data for these types of programs:

- The programs are declared as programs of type P.
- One XRef record is written for each program with the member name as the entry into the program.
- XRef data is written for the file and each field in the file that are accessed in the program using an EXEC ADABAS GENERATE or a COPY statement.

What Types of Code Use are Documented in XRef Data

XRef data is written for the following types of code use:

- CALL and ENTRY statements. For COBOL and PL/I, the Preprocessor scans the source code for these statements.
For Assembler (BAL, 370-Assembler), preprocessor statements have to be included in the source code to identify the call of external programs and functions.
- Use of copy code and include code.
- EXEC CICS statements: All Assembler, COBOL and PL/I programs are scanned for several types of EXEC CICS statements.

The different types are described in the sections below.

Documenting CALL Statements and Entry Points

COBOL or PL/I programs are scanned for CALL statements and entry points.

The following rules apply:

- The name of the entry or external program can be up to 8 characters long.
- In COBOL a maximum amount of 50 entries and 300 called modules are allowed.
- A COBOL entry point is defined by an ENTRY statement.
- A PL/I entry point is defined as the label before a PROC, PROCEDURE or ENTRY statement.
- Equalization of module/entry names in PL/I will be performed. If an external program and an entry are found with both the same name, the existence of an internal procedure will be presumed.
- Assembler programs will be scanned for the macros CALL and CM\$CALL. The names following these statements will be written as the external member names. The entry points and called external programs may also be specified in separate preprocessor statements.

Documenting the Use of Copy Code

All Assembler, COBOL and PL/I programs are scanned for copy code.

- Assembler and COBOL programs are scanned for the COPY statement; the first eight characters of the word following the COPY statement (in the same line) are stored as copy code name.
- PL/I programs are scanned for the INCLUDE statement. All copy/include code names within an INCLUDE statement are stored as XRef data.

Documenting EXEC CICS Statements

All Assembler, COBOL and PL/I programs are scanned for a variety of EXEC CICS statements. The following types of EXEC CICS statements are found and documented:

- EXEC CICS LINK PROGRAM ('literal')
- EXEC CICS XCTL PROGRAM ('literal')
- EXEC CICS LOAD PROGRAM ('literal')
- EXEC CICS SEND MAP ('literal')
- EXEC CICS RECEIVE MAP ('literal')
- EXEC CICS HANDLE ABEND PROGRAM ('literal')

The literal will be written into XRef data as the external member name. If the literal is longer than eight characters, or the brackets contain a variable without apostrophes or quotes, *DYNAMIC will be written.