CA Telon® Application Generator

Data Administration Guide

r5.1

CA Product References

This document references the following CA products:

- CA Telon® Application Generator (CA Telon)
- CA Datacom

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is available on the CA support website at http://ca.com/support.

Contents

Chapter 1: Introduction

CA Telon is a productivity tool designed to simplify and speed development of COBOL and PL/I programs for interactive online and batch applications. It offers an integrated solution to the total online application development process. Yet it remains flexible enough to support design and implementation standards presently in effect at your installation.

A CA Telon-designed online application can consist of one screen or many screens. The screen is the programming unit for the online version of CA Telon. Each screen performs one or more functions and contains field s with edits, data access, mapping, PF key processing, and so on, all defined.

CA Telon allows you to manipulate data and screen flow efficiently. Moreover, it automates the tedious, repetitive aspects of application development thereby reducing errors and generating consistent, high quality, standard code.

CA Telon does not restrict the number of databases or files used by a single screen or CA Telon program. Although you do the majority of your coding with the CA Telon Design Facility, there is no restriction on the amount of custom code that you can insert or the processing that code performs.

CA Telon allows you to accomplish many typical application development projects without custom code. With your own custom code in a CA Telon designed application, there is no task CA Telon cannot solve.

If you can do a task in a native COBOL or PL/I program, you can do it in a CA Telon program. Once created, a screen and its definition can be displayed, revised, or copied at any time. You can export it to card image format for efficient storage under standard library management software.

This guide describes basic data access concepts and administrative tasks. It explains both general administrative tasks for relational databases and administrative tasks specific to several RDBMS methods. In this manual, CA Telon Application Generator, formerly known as CA Telon, is referred to simply as CA Telon.

Note: References to CICS also refer to Transaction Server.

CA Telon Design

CA Telon supports all aspects of application development including systems design prototyping, documentation, and maintenance. CA Telon assists you whether you are creating online or batch programs. The following paragraphs outline only some of the areas where CA Telon can help you.

CA Telon focuses on capturing the external design characteristics of an application and eliminates most of the traditional internal design activities, such as programming specifications.

Many other high level development systems require you to follow their development methodology. But CA Telon's development approach is flexible . If you have an effective system development methodology already in place, you can choose which, if any, CA Telon techniques to use.

Audience

Personnel responsible for CA Telon data administration should use this manual. To use this manual you should have a basic knowledge of environmental considerations (IMS/DC or CICS) and a basic understanding of administrative procedures.

Where to Find Additional Information

You can find additional information about CA Telon in the guides that comprise the CA Telon documentation set. The README.TXT file contained on the documentation CD provides a list of these guides.

Chapter 2: Data Access Concepts

The purpose of Data Administration is to allow the base structure of data for an application to be shared by all programs in the application. All data definitions for an application are maintained in the Data Administration function of the TDF. As you create each program in the application, you include this base structure and modify it for the unique data requirements of each program (read, update, and so on).

CA Telon Data Administration gives the data administrator the ability to make changes in one place that can be "inherited" by many programs. For example, if the key length changes during the development of an application the administrator need only change it in one place versus every program. The impact of this change can also be tracked by using the import or autodoc utilities.

This chapter lays a foundation for working in Data Administration. It introduces you to the concepts and terminology of Data Administration.

Chapters 3-8 of this guide elaborate on data administration concepts for a particular environment or database (for example, CICS and DB2). The sample sessions of Chapters 3 through 8 give examples of how to use most of these concepts.

Using Data Sets

A CA Telon data set defines sequential files and VSAM data sets that can be referenced by CA Telon users during program construction.

A sequential file is a file in which the records are accessed in continuous order. A VSAM (Virtual Storage Access Method) data set is managed by VSAM method services. VSAM data set records can be key-sequenced, entry-sequenced, or arranged by relative record. Key-sequenced VSAM records are placed in the data set in ascending collating sequence by a field, called the key. Entry-sequenced VSAM records are sequenced by the order of their entry in the data set, rather than by a key field. Relative-record data sets consist of a number or fixed-length slots, in which each slot contains a unique relative record number.

CA Telon uses the term data set in a logical sense to refer to either sequential files or VSAM data sets. Each CA Telon data set definition is an independent object identified by the name of the data set. There are no sub-components.

Designers and programmers use the CA Telon data set definition to provide reusability of these objects within CA Telon file groups and programs. Programmers can then access characteristics of a data set on a system-wide basis. You can also specify data set information on a program-by-program basis within each data group of a program.

Data Access

Data access refers to how information from a database is processed by an application program. Data access can be complicated when programming manually. CA Telon automates many of the data access procedures and provides flexible alternatives for environment-specific needs.

Types of Data Access

CA Telon provides two kinds of data access:

- Generated
  - Auto exec
  - User exec
  - SEGEDIT (online programs only)
- Custom

Ways to access data in CA Telon

When you program in native COBOL or PL/I, you need to answer two questions for any input/output (I/O) data access:

- What procedural work and storage statements should be used to execute the I/O?
- Where in the program should the I/O be executed?

The following table illustrates which types of data access answers each need.

Question	Auto Exec	User Exec	SEGEDIT	Custom Data Access
What statement should be used to execute the I/O?	Y	Y	Y	N
Where in the program should the I/O be executed?	Y	N	Y	N

Generated Data Access—Auto Execs

An auto exec (automatically executed access), includes the information necessary to answer both questions in the paragraph labeled "Ways to access data in CA Telon". Each auto exec definition specifies the type of access (for example, read or update) and the key data. The type of auto exec requested determines at what point(s) the generated program executes the I/O.

Generated Data Access—SEGEDITS

The SEGEDIT is a CA Telon function that performs a READ against a database or file to check for the presence of a particular key. Depending on the application user's specification, an error message results if the key is FOUND/NOT FOUND. CA Telon generates the required I/O and positions the code appropriately.

The SEGEDIT answers both questions in the paragraph labeled "Ways to access data in CA Telon." The SEGEDIT statement is a CA Telon statement used to implement an automatic consistency check between screen entry fields and the database or data set data. A file is automatically read to validate the existence of a record. This logic is generated in a pre-defined location in the program.

SEGEDITs are always generated in the same location in the program. Logically they execute after syntax editing of data and before final disposition of data (ADD, UPDATE, and so on).

Custom Data Access

Custom data access uses programmer provided native COBOL or PL/I data access statements at custom code logic points. Computer Associates provides hierarchical structure charts that show the generated structure for a program (refer to the *Programming Concepts Guide*). The standard custom code logic points are shown on these charts.

This type of data access allows more flexibility than generated data access. Custom data access does not include information to answer either of the questions in the paragraph labeled "Ways to access data in CA Telon." Rather, the programmer has complete flexibility in answering these questions for the program.

Choosing Data Access Types

You should always use the most automated form of data access available: auto exec first, then user exec, and finally custom data access, depending upon your application.

SEGEDITs have a specific function in a CA Telon program. They should be used whenever a consistency edit requires a database access. SEGEDITs are a form of AUTOEXEC, unique to consistency data edits.

Generated data access provides easier initial coding and in the long run, easier maintenance. Generated data access is more concise, clear, and standardized and takes advantage of information supplied under Data Administration. It also allows CA Telon to generate system data access documentation, such as the Data Access Summary Report, to assist in the overall understanding of a system.

Auto Execs

Auto execs are the highest level of CA Telon-generated data access. CA Telon generates data access procedural code from non-procedural TELON statements. You can create one auto exec definition per segment, record or row.

Single and Multiple Point Data Access

Auto exec defines the characteristics of the data access and where in the program logic the I/O is executed. For some requests, such as OUTREAD, CA Telon only generates one data access at one point in the program. For other requests, such as UPDATE, CA Telon generates multiple data accesses at multiple points in the program.

Online and Batch Processing

For online processing, auto exec can generate multiple access statements at multiple points in a program and error handling is automatic. For batch processing, the only auto execs available are TRANSACT, MATCH and MERGE. Error handling is automatic for batch as well as online.

Parameters to Define Generated Data Access Statements

The parameters for defining generated data access statements are described in Chapters 3 through 7 in this guide under the subjects "DB2 Access," "DL/I Access," "VSAM Access," and "Queue and Journal Access" respectively. For more details about these parameters, see the Design *Facility Reference Guide*.

Auto Exec Request Types

CA Telon provides many auto exec request types for online and batch processing. These are defined on the Create/Update Data Group screen with the format **AUTOEXEC request**, where **request** is one of the request types shown in this topic.

Online Request Types

CA Telon provides the following auto exec request types for online processing:

- BROWSE
- CREATE
- DELETE
- INREAD
- JOURNAL
- OIREAD
- OUTREAD
- SPBROWSE
- UPDATE

BROWSE

Specify the BROWSE request for a combination of read by key and read next I/O, to be executed during output processing for listing multiple occurrences of a segment, record or row on the screen. The BROWSE is executed in the B-100 section of the program.

CREATE

Specify the CREATE request to create a new segment or record during the processing of input data from the screen. The write is executed at the end of input processing, in the H-100 section of the program.

DELETE

Specify the DELETE request for a combination of a read and a delete during output and input processing to produce the online delete of a segment or record.

The first read is on output processing, during the formatting of output data to the screen. This allows the data being deleted to be displayed on the screen. This read is executed in the A-100 section of the program.

The delete is done at the end of input processing, in the H-100 section. For DL/I, the code generated in H-100 includes a read for update, followed by the actual delete itself.

Note: For queues, CICS only allows deletion of an *entire queue*. There is no mechanism for deleting a single queue record.

INREAD

Specify the INREAD request for a read to be executed during the processing of input data from the screen just prior to processing input field edits. The read is executed in the D-100 section of the program.

JOURNAL

Specify the JOURNAL request to write a journal record to a CICS journal file. It is executed in the H-100 section of the program. JOURNAL is used only for CICS journals. It is the only valid auto exec request for journals.

OIREAD

Specify the OIREAD request for a read to be executed during both output and input processing. The reads are executed in the A-100 and D-100 sections of the program.

OUTREAD

Specify the OUTREAD request for a read to be executed during the formatting of output data to the screen. The read is executed in the A-100 section of the program.

SPBROWSE

Specify the SPBROWSE request for a fetch of a result set originating from a called stored procedure. This will be executed during output processing for a listing multiple occurrences of a segment, record or or row on the screen. The SPBROWSE request in the calling program must be matched by a BROWSE request in the stored procedure itself.

UPDATE

Specify the UPDATE request for a combination of reads and an update during output and input processing to produce the online update of a segment, record, or row.

The first read is on output processing during the formatting of output data to the screen. This allows the data being updated to be displayed on the screen. The read is executed in the A-100 section of the program.

The second read is on input processing, in section D-100, just prior to fields being edited and mapped from the screen input buffer to the segment or record buffer. An update (replace) follows at the end of input processing, in the H-100 section of the program.

DB2 processing

For DB2, CA Telon generates a read in the section D-100 and a replace (EXEC SQL UPDATE) in the section H-100.

DL/I and VSAM processing

For DL/I and VSAM, the first read on input is normally not for update. Since CA Telon cannot guarantee that no additional data access was done to the database or file between the first input read and the update, an additional read for update immediately precedes the actual replace itself, in the H-100 section. This is an extra call, but probably not an extra I/O (depending on the size of the I/O buffer). If this is considered a problem, you should use user exec data access.

CICS temporary storage queues

For CICS temporary storage queues, CA Telon generates a read in D-100, and a replace (EXEC CICS WRITEQ...REWRITE) in section H-100. You cannot update transient data queues, by their very nature.

Batch Request Types

CA Telon provides the following auto exec request types for batch processing:

- MATCHx
- MERGExx
- TRANSACT
- SPTRNACT

MATCHx Request

Specify the batch MATCHx request for sequential reads of a segment, record or row to be used as input to a batch match process. This request is only valid in a program with the MATCH structure. Replace the 'x' in MATCHx with the 'M' for master, or a 'T' for transaction.

MERGExx Request

Specify the batch MERGExx request for sequential reads of a segment, record or row to be used as input to a batch Merge process. This request is only valid in a program with the MERGE structure. Replace the 'xx' in MERGExx with a sequential number-01 through 20.

SPTRNACT Request

Specify the batch SPTRNACT request for a fetch of a result set originating from a called stored procedure. To define transaction input, enter the type in the REQUEST field of the auto exec line for the DB2 table containing the transactions when defining the program data group. The SPTRNACT is executed in the C-1000-GET-TRAN section of the batch program that calls the stored procedure. The SPTRNACT request in the calling program must be matched by a TRANSACT request in the stored procedure itself.

TRANSACT Request

Specify the batch TRANSACT request for sequential reads of a segment, record or row to be used as transaction input to the batch process. To define transaction input, enter the type in the REQUEST field of the auto exec line for the file, DL/I database, or DB2 table containing the transactions when defining the program data group.

The transaction input can be a sequential file, VSAM file, or a sequence of segments of a DL/I database or DB2 table. For a DL/I database, all segments must be of the same type.

Defining Auto Exec Requests

An auto exec statement is defined on the Update Data Group screen by specifying one of the REQUEST types defined above, with a LABEL of AUTOEXEC. If you use one of the REQUEST types that does not have a user exec counterpart (for example, OUTREAD, INREAD, OIREAD, and BROWSE), you can leave the REQUEST field blank and let CA Telon fill in the AUTOEXEC parameter. If you use one of the REQUEST types that has a user exec counterpart (for example, CREATE, UPDATE and DELETE), you must enter a label of AUTOEXEC to identify this statement as an auto exec statement.

The next subject on user execs states that the generated code handles unexpected return codes, and that the IGNORE parameter allows you to handle other return codes in custom code. This is equally true for auto exec requests.

User Exec Request Types

CA Telon provides many user exec request types for more specific processing needs that require flexibility. These are defined on the Create/Update Data Group screen with the format:

- **label request**, where **label** is an optional user exec label
- **request** is one of the request types shown in this topic.

Summary of User Exec Request Types

CA Telon provides the following user exec request types:

- CREATE
- DELETE
- ERASE
- JOURNAL
- READ
- READNEXT
- REPLACE
- UPDATE
- SPRDNEXT

CREATE

Specify the CREATE request to generate a paragraph or procedure to create a new segment, record, or row. The generated code handles unexpected return codes. Expected return codes such as DUPLICATE RECORD are handled in your code which performs or calls the CREATE paragraph or procedure. Expected return codes are any codes specified in the IGNORE parameter.

DELETE

Specify the DELETE request to generate a paragraph or procedure to delete an existing segment, record, or row. The generated code handles unexpected return codes. The only expected return code normally is DELETE OK. For DL/I, the delete includes both a read for update, and a delete. If you have coded a read for update elsewhere and only want a delete, specify a FUNC code of DLET for DL/I on the Update DL/I Detail Data Access screen.

ERASE

Specify the ERASE request to generate a paragraph or procedure to delete an existing segment, record, or row. The generated code handles unexpected return codes. This request functions the same as DELETE, except for DL/I. For DL/I, the generated code includes only a delete, without the read for update generated by a DELETE request. To use ERASE with DL/I, you must execute a read for update (a READ request with a FUNC of GHU, GHN, and so on) prior to the ERASE.

JOURNAL

Specify the JOURNAL request to generate a paragraph or procedure to write a journal record to a CICS journal file. Journal is used only for CICS journals. It is the only valid user exec request for journals.

READ

Specify the READ request to generate a paragraph or procedure to read a segment, record, or row. The generated code handles unexpected return codes. Expected return codes such as RECORD NOT FOUND, are handled in your code which performs or calls the READ paragraph or procedure. Expected return codes are any codes specified in the IGNORE parameter.

READNEXT

Specify the READNEXT request to generate a paragraph or procedure to read an existing segment, record, or row sequentially. You can define a sequential read using the READ user exec statement with an override of the FUNC code. However, since the FUNC code is access method-specific, it must be different for each access method. The generated code handles unexpected return codes. Expected return codes such as RECORD NOT FOUND are handled in your code which performs or calls the READNEXT paragraph or procedure. Expected return codes are any codes specified in the IGNORE parameter.

REPLACE

Specify the REPLACE request to generate a paragraph or procedure to replace an existing segment, record, or row. For DB2, this request functions the same as UPDATE. For DL/I and VSAM, the generated code includes only a replace, without the read for update generated by an UPDATE request. To use REPLACE with DL/I, you must execute a read for update (a READ request with a FUNC of GHU, GHN, and so on) prior to the replace. To use REPLACE with VSAM, you must execute a read for update (a READ request with an OPTLIST value of UPDATE) prior to the REPLACE.

SPRDNEXT

Specify the SPRDNEXT request for a fetch of a result set originating from a called stored procedure. The generated code handles unexpected return codes such as RECORD NOT FOUND are handled in your code which performs or calls the SPRDNEXT paragraph or procedure. Expected return codes are any codes specified in the IGNORE parameter. The SPRDNEXT request in the calling program must be matched by a READNEXT request in the stored procedure in the stored procedure itself.

UPDATE

Specify the UPDATE request to generate a paragraph or procedure to update an existing segment, record, or row. The generated code handles unexpected return codes. Expected return codes such as RECORD NOT FOUND are handled in your code which performs or calls the UPDATE paragraph or procedure. Expected return codes are any codes specified in the IGNORE parameter. For DL/I and VSAM, the update includes both a read for update and a replace.

If you programmed the read for update elsewhere in your code and only want a replace, specify a FUNC code of REPL for DL/I on the Update DL/I Detail Data Access screen or a FUNC code of REWRITE for VSAM on the Update VSAM/SEQ Detail Data Access screen. For DB2, the update includes only a replace (EXEC SQL UPDATE).

Defining User Exec Requests

A user exec statement is defined on the Update Data Group screen by specifying one of the request types defined above, with a label of **label**.

Each of the user exec definitions above generates a COBOL paragraph or a PL/I procedure that is performed or called from code within a custom code logic point in the CA Telon program definition. CA Telon generates the code in the U-100 section of the program in a paragraph or procedure named **U-100-label-segname**, where:

- **label** is the statement label or, if the label is not specified, the statement request (for example, READ, CREATE, or UPDATE).
- **segname** is the CA Telon name of the segment, record, or row being accessed.

If two user execs exist for the same segment, record, or row, and with the same statement type (for example, READ, CREATE, or UPDATE), at least one of them must have the LABEL field defined to avoid duplicate paragraph or procedure names.

An example of custom code to execute a generated paragraph for a user exec READ of a segment, record, or row named TRGEMPL follows.

```
For COBOL:      PERFORM U-100-READ-TRGEMPL.

For PL/I:       CALL U_100_READ_TRGEMPL;
```

The same example, except with a user exec statement LABEL of MYREAD, follows.

```
For COBOL:      PERFORM U-100-MYREAD-TRGEMPL.

For PL/I:       CALL U_100_MYREAD_TRGEMPL;
```

Altering Access

You can alter access by:

- Modifying generated data access statements (see page 22)
- Using generic access definitions (see page 23)

Modifying Generated Data Access Statements

You can easily modify generated data access statements from one access method to another from the Create/Update Data Group screen. For example, if you have an existing application accessing a root-only DL/I database, or a VSAM file, you can alter the access to DB2 with little or no modification to your coding logic. You can also alter a hierarchical DL/I database to DB2, but it probably requires some modification to your coding logic, since you logically treat a hierarchical path of multiple segments somewhat differently from multiple DB2 tables.

To alter the access method for a DL/I database, VSAM file, DB2 table or CICS queue or journal, you simply copy the data access statements from one database, file, or table, to a new database, file, or table that you have added to the program data group.

If you did not use generic access definitions (described next) for the original access, you must check if you had any parameter overrides unique to the old access method, such as requesting a FUNCtion of GHN on a DL/I user exec statement. You must also check for any return code references in the old custom code that must be modified to the new access method.

Using Generic Access Definitions

You can more easily modify the access method if you use generic access definitions when creating the initial program definitions. In fact, if you use generic access definitions throughout an application, you can convert from one access method to another by merely changing the access method type from the Create/Update Data Group screen. This is possible because the generic access auto exec and user exec statements are the same for all access methods and generic access includes a common set of access return codes for all access methods.

Generic access simplifies data access in an environment that uses multiple access methods and also simplifies converting an application from one access method to another. There are two parts to generic access:

- Using only generic access auto exec and user exec definitions (see page 23)
- Using only generic access return codes (see page 24)

Using Only Generic Access Auto Exec and User Exec Definitions

To use generic access for all auto exec and user exec definitions, limit auto exec and user exec parameters to the following list, which can be specified on the Create/Update Data Group screen.

- LABEL
- REQUEST
- KEY/WHERE
- IGNORE

In addition, use a key only without a WHERE specification, for the KEY/WHERE field, and use only generic return codes for the IGNORE field. The valid generic IGNORE parameters are as follows:

IGNORE Parameter	**Description**
OK	Always ignored by CA Telon-generated processing
NOTFOUND/NFD	Not found by access
DUPLICATE/DUP	Multiple record/key occurrences
LOGICERR/LOG	Error on access dependent on prior required condition
SECURITY/SEC	Security violation
ENDFILE/EOF	End of file condition
NOTAVAIL/NAV	Resource not available
DBMERROR/DBM	Any other return code (SQL only)

Using Only Generic Access Return Codes

To use generic access return codes in your custom code, use the following names:

COBOL 88 Levels	**PL/I**
DA-OK	DA_OK
DA-DUPLICATE	DA_DUPLICAT
DA-NOTAVAIL	DA_NOTAVAIL
DA-NOTFOUND	DA_NOTFOUND
DA-ENDFILE	DA_ENDFILE
DA-LOGICERR	DA_LOGICERR

COBOL 88 Levels	PL/I
DA-SECURITY	DA_SECURITY
DA-DBMERROR	DA_DBMERROR (SQL only)
DA-ANYERROR	DA_ANYERROR

Error Handling

CA Telon considers return codes to be one of two types:

- Expected return codes
- Unexpected return codes

For all data accesses except BROWSE, expected return codes include OK (access completed without error) and any return codes defined using the IGNORE parameter. For BROWSE and TRANSACT auto execs, ENDFILE conditions are also valid return codes. When any auto exec or user exec results in an expected return code, CA Telon allows normal execution to continue. If any special processing is necessary for an expected return code, you provide that processing.

When an auto exec and user exec results in an unexpected return code, normal execution is interrupted and CA Telon error handling takes over. The action taken can be customized at your installation.

Multiple I/O Area Handling

Normally you set up a CA Telon program to use only a single I/O area for each segment, record, or row. For some application situations, however, you need to hold multiple records of the same type in the program at the same time. To do this in CA Telon, you must define a second I/O area in the program and request an auto exec or user exec statement to use it.

Defining a Second I/O Area (Method 1)

To define a second I/O area in the program, reference the name of the copy book for the new area as a working storage work area. (You can define this on the Create/Update Screen Definition screen, using the WKAREA parameter.) This causes CA Telon to copy that I/O definition into the working storage section of the program. If this is a duplicate definition of the original I/O area (that is, the element names are the same as those from the original I/O area), the following two steps must be taken:

1. A unique group level name must be given to the new I/O area. Do this by entering the name of a custom code working storage member in the WKAREA parameter. That working storage member has two lines: the group level name, and a COPY statement referencing the original COPY book name.
2. All references to data items in those areas must be fully qualified. For COBOL, this means the variable name referenced for any field on the screen (DBNAME parameter) must use the OF parameter, as well as the DBNAME parameter. For PL/I, this means that each variable name must be preceded by the high-level qualifier.

To specify that an auto exec or user exec statement must use the second I/O area, specify the group level name for the new area in the IOAREA parameter for that auto exec or user exec statement. Do this on the access method detail screen that is accessed from the Create/Update Data Group screen by updating the request.

Defining a Second I/O Area (Method 2)

You can also use the COPY, COPYLV1, and COPYLBL parameters on the Update Data Set Record screen to define a second I/O area.

Inheriting Parameters

Parameter inheritance provides simplicity and flexibility for the application definition process. Simplicity by defining low-level characteristics (parameters) for data access under Data Administration (for example, specifying a key length field). Flexibility, by enabling you to override these parameters at the program definition level. Parameter inheritance also provides simplicity by automatically determining parameters from other information associated with the data access definition.

Definition

An inherited value designated by an **@** in a parameter, means one of the following two things:

- The value of the parameter is taken from a higher level definition. (For example, values set in Data Administration or key values defined at a segment level.)
- The value of the parameter is taken from other information about the access. (For example, a READNEXT request produces a DL/I OP CODE of >= and a PATH CALL produces a DL/I command code of ***D_**.)

Significant @ Meanings

In CA Telon you display the value to be inherited by preceding the value with an @ sign. The following table summarizes significant @ meanings.

Displayed Value	**Description**
@term	Where term is the inherited value. For example, @XFER-ID indicates that XFER-ID is the inherited value. If the displayed value was simply XFER-ID, then the value would be the same, but it would not be inherited.
@	Where @ is followed by spaces. This indicates that the value is to be inherited, but is not shown on the screen. This occurs when the inherited value is the default value.
@?	This signifies that the value from the higher-level definition should have been specified, but was not. This could occur if the higher-level definition is temporarily incomplete. This should be corrected before you generate a CA Telon program from the definition.
@-DSC	(DL/I only) The key field of the Create/Update Data Group screen can sometimes have the value of @-DSC. This indicates that the key field is being inherited, but further indicates that the source of inheritance is a regular DSC (Data Search Criteria) defined in option 2. This is done because you often want to know from the Create/Update Data Group screen that a DSC is being used to specify this key field. In this case, the inherited value cannot be overridden.

Automatic Changing of Inherited Values

If a value is inherited, then it will automatically change if the source of the inherited value changes.

Example 1

If several READ requests were defined in option 4 or 5, each with a value in the IOAREA parameter of @, then the IOAREA value for each READ would change every time the value of IOAREA was changed in option 2.

Example 2

A READ user exec would normally show a DL/I OP CODE of @=, but would show a DL/I OP CODE of @>=, if it were changed to a READNEXT user exec.

You identify that a parameter is to be inherited by typing an @ in the first character of the field. You identify that a parameter is *not* to be inherited by overtyping the @ with the parameter value (you would not need the @ sign). As long as the @ sign is there, it inherits the parameter.

For the first example, specifying an IOAREA value without an @, for any request would cause the specified IOAREA to be used regardless of what changes were made in Option 2.

For the second example, deleting the @ causes = to be used for the OPCODE even if the READ request is changed to a READNEXT request.

Chapter 3: Relational Database Tasks

Before CA Telon will generate an SQL statement in a program to access an SQL table, the definition of the table must be imported into CA Telon from the SQL catalog (or in some cases created by manually adding the definition to the CA Telon Data Administration screen). An auto exec or user exec data access request in a data group for a program definition references an SQL object stored in the CA Telon directory such as a table, a view, or table "join criteria."

You can dynamically import SQL table definitions and views into CA Telon or in some cases you can create them in option 2 (Data Administration). The criteria to join up to 15 tables can also be created in option 2.

CA Telon provides a great amount of flexibility to the user in generating DB2 SQL DML statements. Option 2 is where all of the table definitions needed by programs being generated are stored.

The definition of a table, including all TLNROWs and join criteria specified for it, can be globally used by any number of programs.

This chapter takes you through common CA Telon administrative tasks performed using SQL relational databases.

The first subject, "SQL Access," describes important concepts pertaining to SQL access. The process to list and import SQL tables online is time consuming and resource intensive. To provide more useful functionality to TELON users, a batch SQL import facility has been developed for TELON for each relational database supported. The later subjects give sample sessions for doing SQL tasks, with appropriate explanations.

Install Requirements

SQL tables can be imported into CA Telon TDF Data Administration from the SQL Catalog.

SQL Access

The objective of SQL support in CA Telon is to provide the full flexibility and functionality of SQL access and yet simplify its use and provide compatibility with other access methods used by CA Telon. Toward this end, CA Telon supports SQL with the same auto exec and user exec access statements as used for other access methods.

CA Telon also provides unique SQL features and full support of SQL functions with a combination of SQL parameters and the ability to capture and edit the generated SQL statements.

This subsection gives an overview of SQL access.

SQL Table

A CA Telon SQL Table is a DBMS object that is stored in CA Telon Data Administration and is the equivalent of an SQL table with the particular RDBMS environment. In SQL, a table represents a user entity and is defined using Structured Query Language (SQL).

The concept of a table is similar to the concept of a file or a database segment. An SQL table consists of one or more columns. A column is an attribute of information about the entity, and is equivalent to the concept of a field in a sequential file. For example, a column could represent an employee ID number, employee name, employee department, or employee phone number, and so on.

A CA Telon SQL table consists of two components. It consists of one or more CA Telon rows. A CA Telon row is further comprised of zero or more columns. A column can exist in more than one CA Telon row.

To avoid confusion, it should be pointed out that the term “row” has a different meaning in CA Telon than in SQL. A CA Telon row is a logical subset of the columns in a CA Telon SQL table or a CA Telon joined table. A CA Telon row equates fairly well to the concept of an SQL table view.

The specification of CA Telon rows is an important part of defining an SQL table. CA Telon rows serve as reusable SELECT columns clauses when table accesses are made against them in a program's data access definition.

Referencing a CA Telon row name in a SELECT clause instead of listing all the columns can improve a program's productivity. It removes the need to key in all the column names, and reduces the time needed to resolve errors that result from mistyping column names. It can also help a programmer compensate for a lack of knowledge about a particular SQL table.

CA Telon supports dynamic inheritance of all column names with the CA Telon row into the SELECT clause of the generated SELECT statements.

Joined Table

A CA Telon joined table is a special feature provided by CA Telon to support SQL data access. Defining and using CA Telon joined tables can simplify the processing of a join of multiple SQL tables. SQL tables.

A CA Telon joined table can be thought of as the conceptual table that results from the **execution of an SQL join query.** A CA Telon joined table is not a regular DBMS object, such as an SQL table, and does not exist outside of CA Telon. A CA Telon row is similar in function to an SQL table view.

Components of a CA Telon Join Query

It is important to understand the main components of an SQL join query in order to understand the components of a CA Telon joined table. An SQL join query consists of a SELECT statement listing the access columns with a FROM clause identifying two or more SQL tables, and a WHERE clause containing the join predicate (join where conditions). This criteria is used to define CA Telon joins. An example of a simple join query with two participating tables follows:

SELECT	table1.column1,table1.column2,
	table1.column3,table2,column1,
	table2.column2
FROM	table1,table2
WHERE	table1.column3 = table2.column3

A CA Telon joined table consists of two to fifteen joins, one to many join WHERE conditions, and one CA Telon row comprised of one to many access columns.

Join Tables

A CA Telon joined table cannot itself participate in a join. You can list an SQL table as participating in a join more than once (for example, a table can be joined to itself). This list of tables forms the FROM clause in the SQL SELECT statement.

Join Where Conditions

The join WHERE conditions are ANDed together to form the join predicate in the SQL join query. The join predicate comprises the foundation WHERE clause in the SQL SELECT statement. The WHERE clause can be modified in a program's data access group.

CA Telon Row

CA Telon creates a default CA Telon row is when a CA Telon joined table is defined. Its row name must be the same as the CA Telon table name.

A CA Telon joined table can only have one CA Telon row, comprised of all the access columns. When a CA Telon joined table is referenced in a program's data group, the CA Telon row component is visible and its use is more apparent.

Access Columns

When a join is DGADDED to a program, all the columns from the first CA Telon row of each of the SQL tables participating in the join are available to the program via inheritance. Once the access columns are inherited, you can alter them as you desire.

Data Administration Functions

Special functions exist in CA Telon option 2 to assist you in defining auto exec and user exec access statements. These are listed below:

- DB2 catalog access—Allows you to list entries from the DB2 catalog directly from Data Administration. From that list you can request tables to be imported immediately into CA Telon.

  Note: This direct access to an SQL catalog is only available for DB2 and only with the mainframe TDF.

- Alias definition—Allows you to define COBOL or PL/I aliases for table column names. Aliases can make host variable names more meaningful and eliminate duplicate names and the level of qualification that can result. Aliases are defined on the Create/Update SQL Tables/Rows screen.

- Key definition—Allows you to generate SELECT statements based on the value of one or more columns. Key fields are also defined on the Create/Update SQL Tables/Rows screen. As explained under Key Fields and Indices below, you can identify columns as key fields.

- TLNROWs—Allows you to create column subsets of an SQL table for use in shorter access statements similar to SQL views. Its use is explained under SQL Column Selects (see page 33). You create a TLNROW on the Create/Update SQL Tables/Rows screen by:

  - Repeating the table entries, including the TLNROW line
  - Deleting any columns not to be included in the new TLNROW
  - Entering a new TLNROW name in the ALIAS column for the new TLNROW line.

- Table Joins—Allows you to define the join criteria of up to 15 tables, giving the join a name, and then using the join criteria with auto execs and user execs in a screen definition or batch definition data group. A join is created from the Data Administration menu. Its use is explained below under table joins (see page 36).

SQL Column Selects

CA Telon simplifies the selection of SQL columns by allowing you to perform the following:

- Define a subset of columns for a table as a TLNROW (CA Telon row) in CA Telon option 2. If the name of this TLNROW is used instead of the table name in the auto exec or user exec, the generated SQL statements list the columns defined in the TLNROW, rather than of all the columns in the table.
- If you request an auto exec or user exec for an SQL table without using the SENCOLS parameter on the Update SQL Detail Data Access screen, the generated SQL statements list all columns in the table being accessed. This is the normal situation.
- Select individual column names from a list of names from the table being accessed. If you enter a non-blank character in the rightmost field for the SENCOLS parameter on the Update SQL Detail Data Access screen, you are taken to a list of column names for the table being accessed. Enter an **S** beside the column names you want included in the generated SQL statement.

Table Size Considerations

In considering accesses against an SQL table, you need to consider two groups of tables: large tables and small tables.

- Large tables are those that are too large to be brought into memory at one time. Performance dictates that you access only a few rows at a time. Indices for these tables are required, and online access of any row is not feasible except through an index. These tables look very much like classic files or databases and are accessed in much the same manner.
- A small table is a table small enough that a large portion of the table can be brought into memory at one time. No indices are required for tables of this size and you get adequate performance when you search for a row or group of rows by any column in the table.

For auto exec and user exec access requests, you need to consider the size of the table you are accessing according to its size. If you treat a large table as if it were a small table, performance can suffer. On the other hand, if you treat a small table as a large table, you lose much of the flexibility that makes SQL attractive.

These considerations are not CA Telon-specific. They are factors that must be considered when accessing SQL tables through any online transaction driven system, such as IMS/DC and CICS.

Key Fields and Indices

Large tables should be accessed only through inedex columns. CA Telon assists you in controlling this restriction by carrying the concept of keys into SQL. When an SQL table is defined to CA Telon through CA Telon option 2, you have the option of defining some columns as key fields. You should select each column defined in the prime index path as a key field.

When you set up columns as key fields, accesses against that table do not require the specification of both the SQL column name and the host variable name including the select criteria. Rather, they only require the host variable name. This saves you from having to remember the SQL column names. More importantly, however, it removes the risk of accidentally selecting against columns that are not index entries.

You can implement alternate indexes into a table by creating multiple TLNROWs for a table and specifying key fields in each TLNROW that defines an access path. An auto exec or user exec which is to access the table through a particular index path can then simply reference the TLNROW name for which the correct path is defined.

If a table with an index defined for it is imported into CA Telon, a separate TLNROW is automatically created for the index.

A final advantage in setting up key fields is that SQL accesses can be treated the same as other file and database accesses. This lowers the learning curve when SQL tables and files or databases are included in the same application.

List Screen Considerations (Online Only)

Processing list screens from data in SQL tables is very similar to processing list screens from data in other databases and files. Yet there are also some special considerations for accessing SQL tables.

Table size

The first consideration addresses table size. When list screens access other databases and files, CA Telon only reads enough segments or records to fill up one screen of data. One reason for doing this is consistency of the application user paging interface. The main reason for doing this, however, is performance. One of the performance problems to be avoided in transaction driven systems, like CICS and IMS/DC, is accessing large numbers of segments or records at a time. This often occurs for list type processing where a large number of data items are to be presented to the operator.

This same performance problem exists for SQL, when accessing data in very large tables. As noted earlier, online access of such large tables is only feasible from a performance standpoint when an index is used. When accessing through that index, CA Telon formats only one page at a time, returning control to the operator after each page is displayed. As with other databases and files, this eliminates the performance problem of large numbers of accesses in one transaction.

Small tables

The considerations are very different, however, for small tables. This is especially true when the table is not accessed through an index. In this case, the technique that assists performance for large tables actually degrades performance.

Without access through an index, the SQL RDBMS is likely to read a large number of physical records to create the first page. Even worse, those same records plus additional records are accessed on the second page.

Each page forward causes the reread of all data from prior pages. To avoid this problem, small tables, especially those without indices, can be read into an array in the transfer work area. Specifically, this involves using a temporary storage technique where you use custom code in OINIT1 or OINIT2 and in OCUST1 or OCUST3. You can then use SEGLOOP with table mapping to map the data from the array to the screen for each page.

Multi-Part keys

Start Browse (STBRKEY) functions the same for SQL as for other databases and files. A condition that occurs more often, however, is the existence of an SQL table with a multi-part key (that is, more than one column identifies each row within the SQL index for that table). For this situation, you must set up the field for the STBRKEY value as a concatenated field for all columns in working storage or the transfer work area.

For example, if the index to a table contains three columns of lengths 4, 3 and 6, then you must set up a field of 13 bytes and move the key values of all three columns into that concatenated key.

The WHERE feature

The WHERE feature of SQL provides some additional capability when creating a selected list of data items. In other words, when you do not want to list all the rows that normally are presented when accessing sequentially through the index, you can use the WHERE clause to access only the rows you need. For this situation, you set up a WHERE clause on the BROWSE auto exec statement, selecting only the desired rows. The list proceeds sequentially through the SQL index but only returns rows that meet the selection criteria.

You can use both a STBRKEY and a WHERE clause on the same SEGLOOP screen.

Table Joins

It is often desirable to treat a join of multiple tables as if it were an SQL table set up in Data Administration. CA Telon simplifies join processing by allowing an SQL join to be defined in CA Telon option 2. It provides a number of screens, described in the *Design Facility Reference Guide* that allow you to define the criteria for the join. To a CA Telon developer a CA Telon join looks as if it were a regular SQL table.

The SQL statements generated by CA Telon contain all the parameters necessary to join the tables according to the criteria defined in option 2.

Sequential Row Processing

The basic concepts behind relational databases do not include the idea of sequential processing. You are to treat everything as tables and make selection requests that only make new tables from old tables. This works quite well for query applications, but not for production systems, when very large tables must be accessed efficiently through transaction-driven systems like CICS and IMS/DC. It is also a problem for programming languages like COBOL and PL/I, for which handling unlimited arrays is difficult.

Because of the above problems, SQL allows the concepts of sequential processing for COBOL and PL/I. One form of sequential processing, that is used for list screens, has been covered above. An additional form of sequential processing is implemented through the user exec READNEXT statement, as well as the related SPRDNEXT statement for programs that call stored procedures.

The general considerations of READNEXT and SPRDNEXT data access are described under User Exec in the "Data Access Concepts" chapter. There are no special considerations in reading rows from small tables into arrays in working storage. There are special considerations, however, when processing large tables sequentially in a prescribed order. For these situations, an SQL index should be defined for any column that is used in ordering a large table. Other considerations are discussed below.

A READNEXT user exec generates a paragraph or procedure with a DECLARE CURSOR statement and a FETCH statement. The READNEXT user exec performs the same function (SQL statements) as an auto exec BROWSE. One execution of the paragraph or procedure returns a single row to the program. The paragraph or procedure is performed or called multiple times to return a series of rows, in the order of the SQL index if the WHERE clause specifies the index.

The WHERE clause from the READNEXT is used for two purposes:

- For indexed tables, it can identify a position within the SQL index at which the first sequential read takes place (a start browse condition). After that position, the rows are then presented one after another.
- It can act as a "filter" so that only rows meeting certain criteria are returned.

In some sequential processing, you want to start at a certain logical point within the database and proceed forward sequentially. This is sometimes referred to as browsing the database with a start browse key. Using the WHERE clause as a "start browse" condition works well when the WHERE clause from the READNEXT identifies one or more columns that are part of the index. It may not work otherwise.

Without accessing through an index, you cannot even be sure that the rows are presented in the same order for two separate executions. Even worse, performance is significantly degraded for large tables. Therefore, for large tables, you should only use sequential processing when ordering by columns contained in an SQL index.

You also need READNEXT processing for any reads where multiple rows meet the search criteria. Normally a problem occurs since the SQL SELECT statement in COBOL or PL/I generated by CA Telon from a READ request cannot return more than one row at a time.

If you issue an SQL SELECT statement when multiple rows meet the selection criteria, SQL returns an error code, without returning any row. To handle this problem, the program needs to use a DECLARE CURSOR statement, followed by a FETCH request for each row. CA Telon generates the DECLARE CURSOR and FETCH combination from a READNEXT (or BROWSE) request.

Therefore, you must use a READNEXT user exec rather than a READ user exec when accessing a table with selection criteria that is not unique.

The above problem usually occurs when there are duplicate keys. To minimize this problem, make your selection criteria in a READ request (the KEY/WHERE field in a CA Telon auto exec or user exec) extensive enough to be unique.

For the SPRDNEXT data access there are several major differences to what is generated for READNEXT. In the first place, a SPRDNEXT data access should only be used in a program which calls a stored procedure which itself contains a matching READNEXT, containing a HOLDCUR=Y specification. Then the functionality of the sequential row processing is divided between the stored procedure and the calling program.

In the stored procedure, the DECLARE CURSOR is generated, as well as the OPEN CURSOR. When this statement is executed, a result set will be created to ultimately be processed by the calling program.

Note: The CLOSE CURSOR command is not executed in the stored procedure.

Then in the calling program, the SPRDNEXT specification entails a FETCH of the result set, with the CLOSE CURSOR being executed if an SQLCODE of +100 is found from the FETCH. Hence the READNEXT in the stored procedure and the SPRDNEXT in the calling program work together to provide essentially the same functionality as the READNEXT in a program which does not interface with a stored procedure.

Previewing Generated SQL Statements

If you understand SQL and want to review the SQL statements that are generated from an auto exec or user exec, or if you are interested in learning SQL and want to review CA Telon-generated SQL statements as examples, you can use the Preview facility provided as a part of CA Telon SQL support. This facility is available when you are entering or updating generated data access statements on either the Create/Update Data Group screen or the Update SQL Detail Data Access screen.

Use the Preview facility by requesting a preview of any auto exec or user exec statement from one of these two screens. You make the request by entering the characters **VIEW** (or **V**) on the auto exec or user exec line on the Create/Update Data Group screen or by entering any (non-blank) character in the PREVIEW option on the Update SQL Detail Data Access screen. CA Telon then displays a screen containing the SQL statements that are generated in your program from that auto exec or user exec.

To understand the effect of different CA Telon parameters on the generated code, return to the prior CA Telon screen and modify one or more parameters. Then preview the generated code again to see its effect. As explained next, you can also choose to edit the previewed code and request that CA Telon use that code instead of generating it from the CA Telon auto exec or user exec statement.

Inserting Code into Generated SQL Statements

CA Telon allows you to insert code at points in the generated statement to provide additional flexibility. For example, you can include code to expand encoded fields for use by the program. If this code is included as a part of the paragraph or procedure generated from the user exec, it is automatically executed each time the user exec is executed.

Be careful, however, not to overuse this technique. If you start to include programming logic within the user exec, your program can become difficult to maintain. Only use this facility to add flexibility to the I/O request, not to add general programming logic.

You can enhance the use of this facility by setting up any special accesses, such as above, in an application BASE. When a new program definition is created that references the BASE, the auto exec and user exec statements, with custom code members, are included in the new program definition.

CA Telon provides the following custom code points in which you can insert code in the generated SQL paragraph or procedure. Note that the COPY member can either be a part of the CA Telon program definition (in which case you can update it from the Update SQL Detail Data Access screen) or it can be a member in an external library.

- CYPCALL—For modifying the whole generated statement by identifying a COPY member, as explained under "Customizing Generated SQL Statements" described next
- CPYKEY—For replacing the WHERE clause in a generated statement
- CPYINIT—For inserting code at the beginning of the generated paragraph or procedure by identifying a COPY member containing that code
- CPYTERM—For inserting code at the end of the generated paragraph or procedure by identifying a COPY member containing that code

Customizing Generated SQL Statements

CA Telon provides you with considerable flexibility in generating DB2 SQL statements. Sometimes, however, you need the full power of manipulating SQL statements yourself to perform the necessary function. Unfortunately, SQL programming is usually complex, verbose and error-prone in these cases.

CA Telon assists you in coding these statements by generating statements that are close to what you need, and then allowing you to modify these statements using the CA Telon Editor.

Begin the process by setting up a user exec that comes as close as possible to the function you need. You can use the Preview facility described previously to review the generated code to see how close it comes to meeting your requirements.

Then from the Update SQL Detail Data Access screen, select the CPYCALL option. This generates the SQL statements from the user exec parameters and puts you in CA Telon Edit mode so that you can modify the SQL statements.

You can select the option with or without a member name. If you enter a member name, the code you just edited is stored as a custom code member under the current program definition with that name. If you do not enter a member name, the code you just edited is stored under the current program definition with the name CPYCALL. You should use a member name to avoid duplicate members named CPYCALL.

Advantages of using this feature rather than coding the SQL statements from scratch are:

- You save the coding time and resulting errors that are a part of SQL programming
- The modified code is tied to a user exec statement, which can appear on CA Telon management reports (as can auto exec statements).

You should *only* use this feature *when absolutely necessary*. Maintenance is much easier when the SQL statements are generated from the high level user exec.

Importing TABLE Definitions from the SQL Catalog

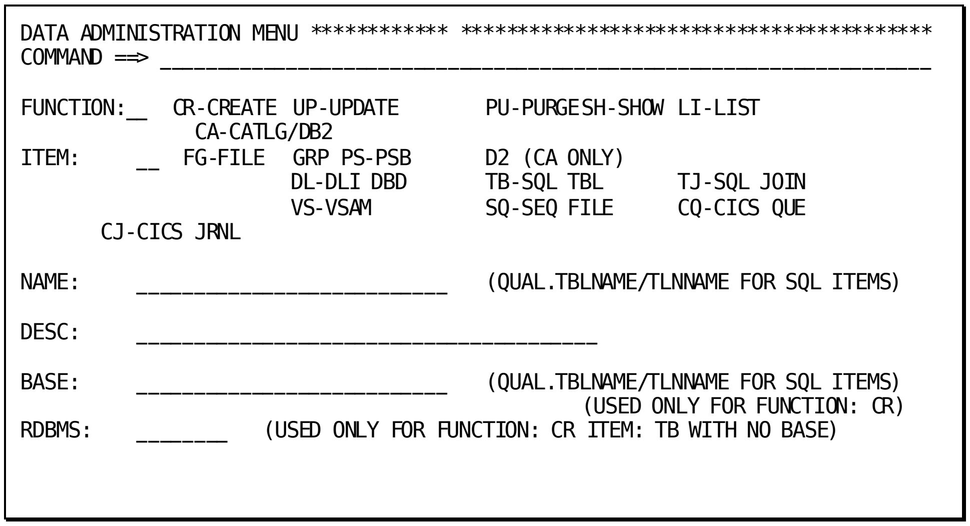
The first task for data administration is to import the TABLE definitions from the SQL catalog. CA Telon allows you to accomplish this task online for DB2 only, and in batch for all supported relational databases. For DB2, see chapter 7 for details of the Batch Catalog Import and SQL Catalog extract. For other relational databases, see the appropriate target manual.

Showing TABLE and TABLE JOIN Definitions for CA Telon

This topic tells you how to display existing table and table join definitions.

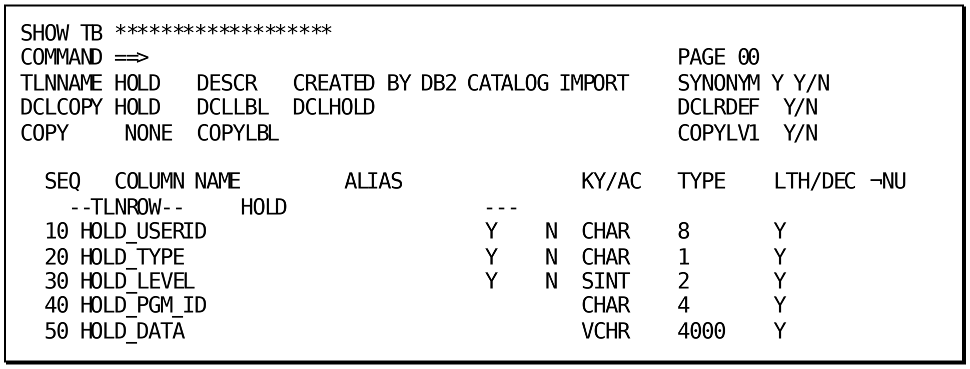
Showing TABLE Definitions

On the Data Administration Menu, type **SH** in the FUNCTION field, **TB** in the ITEM field, the name of the SQL table in the NAME field (in this case, TELON.HOLD), and press Enter.



```
DATA ADMINISTRATION MENU ************ ****************************************
COMMAND ==> __________________________________________________________________

FUNCTION:__   CR-CREATE  UP-UPDATE       PU-PURGESH-SHOW  LI-LIST
                CA-CATLG/DB2
ITEM:      __  FG-FILE   GRP PS-PSB      D2 (CA ONLY)
                        DL-DLI DBD       TB-SQL TBL       TJ-SQL JOIN
                        VS-VSAM          SQ-SEQ FILE      CQ-CICS QUE
       CJ-CICS JRNL

NAME:      __________________________   (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)

DESC:      _______________________________________

BASE:      __________________________   (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                                (USED ONLY FOR FUNCTION: CR)
RDBMS:     ________   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Show TB screen.



```
SHOW TB ******************
COMMAND ==>                                             PAGE 00
TLNNAME HOLD   DESCR   CREATED BY DB2 CATALOG IMPORT    SYNONYM Y Y/N
DCLCOPY HOLD   DCLLBL  DCLHOLD                          DCLRDEF  Y/N
COPY     NONE  COPYLBL                                  COPYLV1  Y/N

  SEQ   COLUMN NAME        ALIAS              KY/AC    TYPE    LTH/DEC ¬NU
    --TLNROW--     HOLD                   ---
  10 HOLD_USERID                          Y    N  CHAR     8       Y
  20 HOLD_TYPE                            Y    N  CHAR     1       Y
  30 HOLD_LEVEL                           Y    N  SINT     2       Y
  40 HOLD_PGM_ID                                  CHAR     4       Y
  50 HOLD_DATA                                    VCHR     4000    Y
```

The following information is provided for the selected table:

- TLNNAME/DESCR—Table name and description.
- SYNONYM—Specifies whether CA Telon is to generate the qualifier for the table.
- DCLCOPY—Name that CA Telon gives in an EXEC SQL INCLUDE (for DB2, this is the SQL DCLGEN COPY member name).
- DCLLBL—Name for the COBOL or PL/I structure that the DCLGEN produces for this table.
- DCLRDEF—Specifies whether CA Telon is to redefine the DCLGEN INCLUDE structure with the COPY or INCLUDE member specified in the COPY field.
- COPY—The COPY/INCLUDE member that CA Telon is to copy into working storage for the table. This member name is the member name for the ALIASes.

  A value of NONE prevents the copy.
- COPYLBL—COBOL data item name of the group level for the Table I/O area copy definition.
- COPYLV1—Indicates whether the COBOL COPY or PL/I INCLUDE member specified in the COPY field begins with a 01—level data item.
- TLNROW—Indicates a TLNROW line.
- SEQ/COLUMN NAME—Sequence number and column names within each TLNROW.
- ALIAS—Name of the TLNROW or the I/O area or host variable name for the column.
- KY/AC—Indicates whether this column is a key column (KY) and whether the column is accessed in generated SQL statements (AC).
- TYPE—Type of fields in the table. (See the section "Create/Update SQL Tables/TLNROWS" of the Design Facility Reference Guide for a list and description of the field types.)
- LTH—Length, if the type is variable length, or precision, if the type is DECIMAL.
- DEC—Scale value, if the type is DECIMAL.
- NU—Indicates whether null values are allowed in the column.

  Note: Fields on the SHOW TB screen are protected. If you want to change information, use the Update SQL Table screen (for more information, see the section Updating SQL TABLE or TLNROW).

Showing TABLE JOIN Definitions

On the Data Administration Menu, type **SH** in the FUNCTION field, **TJ** in the ITEM field, the name of the SQL table in the NAME field (in this case, TELON.TJOIN), and press Enter.

```
DATA ADMINISTRATION MENU ************ *****************************************
COMMAND ==> ____________________________________________________________________

FUNCTION:__  CR-CREATE  UP-UPDATE        PU-PURGESH-SHOW  LI-LIST
               CA-CATLG/DB2
ITEM:     __  FG-FILE    GRP  PS-PSB        D2 (CA ONLY)
                         DL-DLI DBD         TB-SQL TBL     TJ-SQL JOIN
                         VS-VSAM            SQ-SEQ FILE    CQ-CICS QUE    CJ-CICS JRNL

NAME:     ___________________________       (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)

DESC:     ________________________________________

BASE:     ___________________________       (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                                    (USED ONLY FOR FUNCTION: CR)

RDBMS:    ________   USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the SHOW TJ screen.

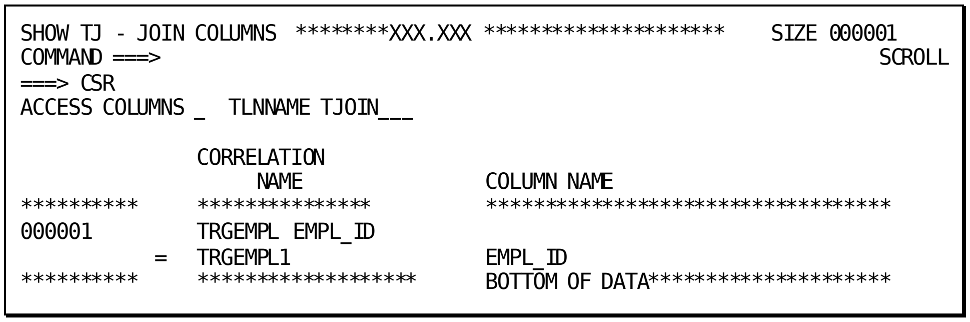
```
 SHOW TJ **************************** *****************************************
 COMMAND ==> ___________________________________________________________________
 JOIN NAME TELON.JOIN
 TLNNAME  TJOIN

  CORRELATION
    NAME       QUAL       TABLE NAME          TLNNAME    SQL TYPE    SQL SCHEMA
   ********   ********   ******************   ********   ********   ************
1  TRGEMPL_   TELON___   TRGEMPL__________    TRGEMPL_   DB2____    ____________
2  TRGEMPL1   TELON___   TRGEMPL1_________    TRGEMPL1   DB2____    ____________
3  ________   ________   __________________   ________   _______    ____________
4  ________   ________   __________________   ________   _______    ____________
5  ________   ________   __________________   ________   _______    ____________
6  ________   ________   __________________   ________   _______    ____________
7  ________   ________   __________________   ________   _______    ____________
8  ________   ________   __________________   ________   _______    ____________
9  ________   ________   __________________   ________   _______    ____________
10 ________   ________   __________________   ________   _______    ____________
11 ________   ________   __________________   ________   _______    ____________
12 ________   ________   __________________   ________   _______    ____________
13 ________   ________   __________________   ________   _______    ____________
14 ________   ________   __________________   ________   _______    ____________
15 ________   ________   __________________   ________   _______    ____________
```

The following information is provided for existing join tables:

- JOINNAME—Qualified join name.
- TLNNAME—Unique CA Telon name for the join.
- CORRELATION NAME—SQL correlation name for the table in the generated auto exec (user I/O) for this join.
- QUAL—Qualifier of the table used in this join.
- TABLE NAME—Name of the table used in this join.
- TLNNAME—CA Telon name of the table used in this join.
- SQL TYPE—SQL table type.
- SQL SCHEMA—Identifies CA Datacom Database SQL schema name of the table. This field is displayed only for CA Datacom Database SQL tables in the join.

  Note: Fields on the SHOW TJ and SHOW TJ-JOIN COLUMNS (discussednext) screens are protected. See the section "Creating TABLE JOIN Criteria" for details about (creating and) updating table join information.

Press PF3 after viewing the SHOW TJ screen. CA Telon displays the SHOW TJ-JOIN COLUMNS screen.



```
SHOW TJ - JOIN COLUMNS  ********XXX.XXX ********************    SIZE 000001
COMMAND ===>                                                          SCROLL
===> CSR
ACCESS COLUMNS _  TLNNAME TJOIN___

               CORRELATION
                    NAME                 COLUMN NAME
**********     ***************           **********************************
000001         TRGEMPL  EMPL_ID
           =   TRGEMPL1                  EMPL_ID
**********     *******************       BOTTOM OF DATA********************
```

The following information is provided:

- ACCESS COLUMNS—Always blank on the SHOW TJ—JOIN COLUMNS screen.
- TLNNAME—CA Telon name the join used as the key name for the join.
- CORRELATION NAME—Correlation names of the tables being joined.
- COLUMN—Names of the columns on which the tables are being joined.

Creating TABLE Definitions

You need to create TABLE definitions for CA Telon when you know what an SQL table will consist of, but it has not been created in the SQL catalog yet. In this way, programs can be designed and prototyped before the actual tables exist.

If the table already exists in the SQL catalog, it is recommended that the import option be used rather than create. This ensures that the information in CA Telon about the table matches what is in the SQL catalog. The SQL statements generated by CA Telon in a program will then be accurate as well.

You can create table definitions either from scratch or by copying an existing table. Both methods are described on the following pages.

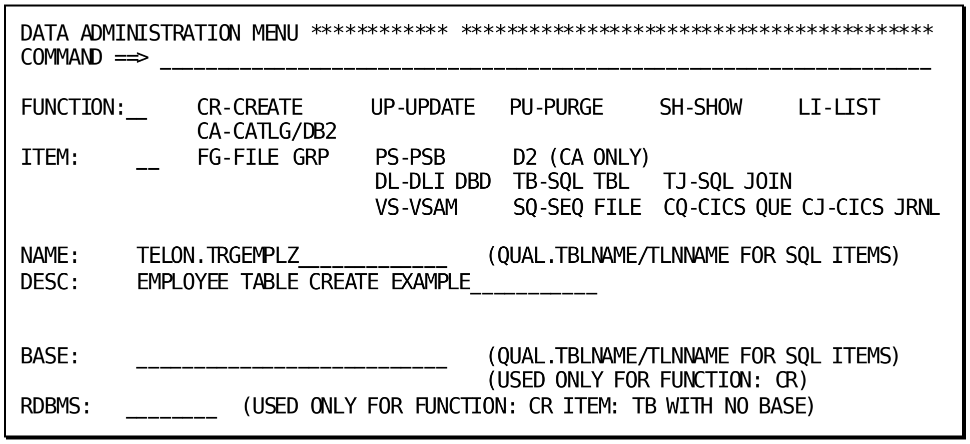
Note: The ability to create TABLE definitions may be suppressed for certain relational databases, especially when adherence to the SQL catalog is mandatory. Check with individual target documentation for details.

Creating a Table Definition from Scratch

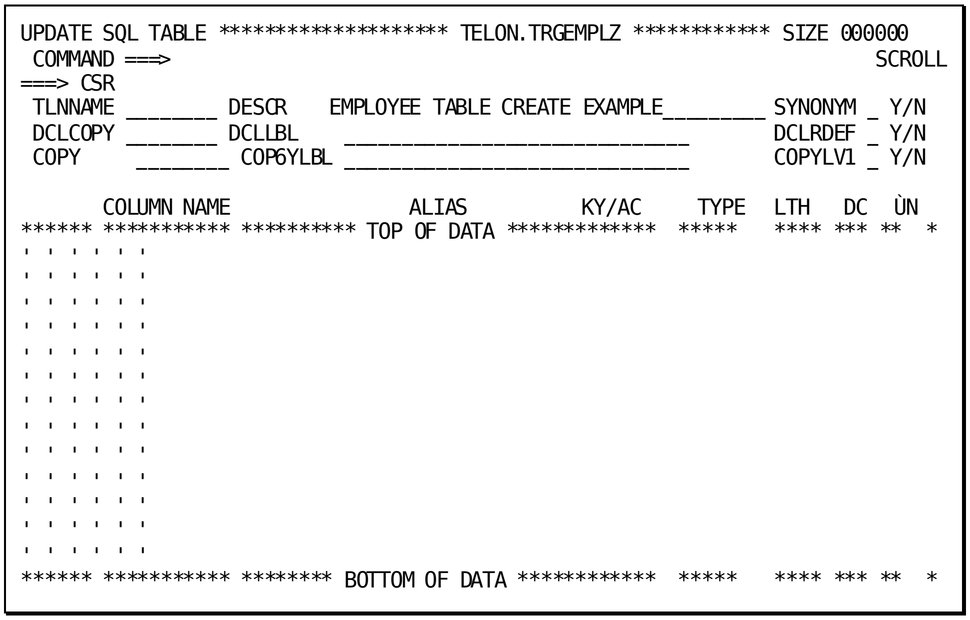
This topic shows you how to create new SQL Table Definitions for CA Telon.

On the Data Administration menu, type **CR** in the FUNCTION field, **TB** in the ITEM field, the name of the SQL table in the NAME field (in this case, **TELON.TRGEMPLZ**), a description in the DESC field, and press Enter. This takes you to the Update SQL Table screen.

Note: The SQL name must be unique and you must enter a description.


```
DATA ADMINISTRATION MENU ************ ****************************************
COMMAND ==> __________________________________________________________________

FUNCTION:__    CR-CREATE      UP-UPDATE   PU-PURGE     SH-SHOW     LI-LIST
               CA-CATLG/DB2
ITEM:     __   FG-FILE GRP    PS-PSB      D2 (CA ONLY)
                              DL-DLI DBD  TB-SQL TBL   TJ-SQL JOIN
                              VS-VSAM     SQ-SEQ FILE  CQ-CICS QUE CJ-CICS JRNL

NAME:     TELON.TRGEMPLZ____________   (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
DESC:     EMPLOYEE TABLE CREATE EXAMPLE___________


BASE:     __________________________   (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)
RDBMS:   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

When you press Enter from this Data Administration Menu, the following Create SQL Table screen appears:



```
UPDATE SQL TABLE ******************** TELON.TRGEMPLZ ************ SIZE 000000
 COMMAND ===>                                                          SCROLL
===> CSR
 TLNNAME ________ DESCR    EMPLOYEE TABLE CREATE EXAMPLE_________ SYNONYM _ Y/N
 DCLCOPY ________ DCLLBL   ______________________________         DCLRDEF _ Y/N
 COPY     ________ COP6YLBL ______________________________         COPYLV1 _ Y/N

       COLUMN NAME               ALIAS          KY/AC     TYPE  LTH   DC  ÙN
****** *********** ********** TOP OF DATA *************  *****   **** *** **  *
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
****** *********** ******** BOTTOM OF DATA ************  *****   **** *** **  *
```

The Create SQL Table screen is used to manually enter all the appropriate information about the SQL table that would ordinarily be supplied by an import. In a sense, you are describing to CA Telon what the DCLGEN for the table will look like. But there is no direct relationship between a DCLGEN and option 2 in CA Telon. The relationship is between the DB2 DCLGEN COPY member name and the generated code.

The next Create SQL Table screen shows sample input:

```
UPDATE SQL TABLE ******************* TELON.TRGEMPLZ ************ SIZE 000000
COMMAND ===>                          SCROLL ===> CSR
TLNNAME  TRGEMPLZ  DESCR  EMPLOYEE TABLE CREATE EXAMPLE_____ SYNONYM    _ Y/N
DCLCOPY  TRGEMPLZ  DCLLBL  ____________________________
      DCLRDEF _ Y/N
COPY       NONE____   COPYLBL  ____________________________
      COPYLV1 _ Y/N


      COLUMN NAME             ALIAS                  KY/AC   TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****   **** *** **  *
' ' ' ' ' '   --TLNROW--     TRGEMPLZ
' ' ' ' ' '   EMPL_ID       @EMPL-ID                 Y       CHAR 6        Y
' ' ' ' ' '   EMPL_NAME   @EMPL-NAME                         CHAR 25       Y


' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
' ' ' ' ' '
****** *********** ******** BOTTOM OF DATA ************  *****   **** *** **  *
```

Note: This exhibit does not show all data typed, but only two entries. For more information, refer to *the Design Facility Reference* Guide.

Enter the TLNNAME parameter (in this case, **TRGEMPLZ**) to specify the name for a table. Enter the DCLCOPY parameter (in this case, **TRGEMPLZ**) to specify the name that CA Telon gives an EXEC SQL INCLUDE for the DCLGEN COPY member name.

Enter **NONE** in the COPY field to prevent a COBOL copy statement from being generated for a copy member containing the definition of the table. You cannot use both COPY and DCLCOPY; they are mutually exclusive.

In the COLUMN NAME field, specify the literal --**TLNROW**-- to identify a TLNROW line. Then continue to enter your column names (only two shown below).

Specify the ALIAS field name (the COBOL or PL/I host variables). These fields designate the name of the TLNROW you are defining. CA Telon uses the name in the generated I/O procedure or paragraph names.

Enter a **Y** in the KY column if this column is to be treated by CA Telon as a key column when generating a WHERE clause in an SQL statement. Use the AC (Access) column if this column is accessed in generated SQL statements.

Typically, option 2 defines all of the columns in a TLNROW as accessible ("YA" is the default). In a particular program definition's data group, the SENCOLs parameter can be used to limit which columns in the TLNROW will be accessed. You can also use multiple TLNROWs for this purpose.

Use the TYPE and LTH fields to specify the type and length of the column. Use DEC to indicate the number of significant digits in a DECIMAL column type.

Specify a **Y** in the ˆNU (NOTNULL) parameter to indicate that nulls are permitted.

Note: When creating an Oracle or CA Datacom Database table from scratch, the RDBMS field on the Data Administration Menu must specify the desired RDBMS. If the field is left blank, the system assumes that the RDBMS is DB2.

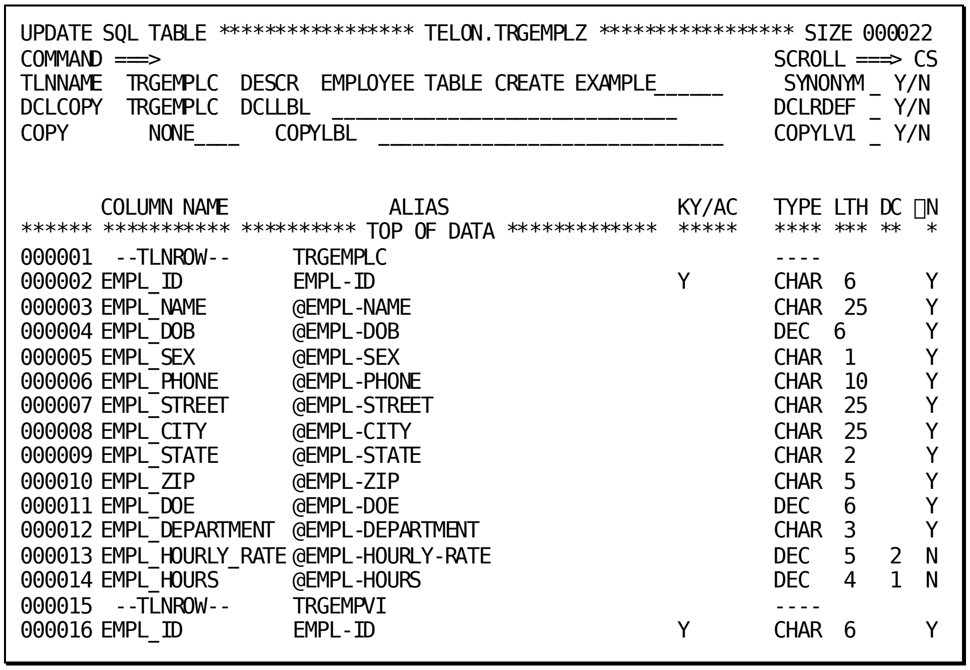
Copying an Existing Table Definition

Another way to create TABLE definitions is to use an existing TABLE definition as a base.

The input on the Data Administration menu is the same as Scenario 1 except that you type an existing SQL table name in the BASE field (in this case, **TELON.TRGEMPLC**). You can now update the new table definition, change the TLNNAME, and do anything else that is unique to the new definition.

```
DATA ADMINISTRATION MENU ************ ****************************************
COMMAND ==> __________________________________________________________________

FUNCTION:CR    CR-CREATE      UP-UPDATE   PU-PURGE     SH-SHOW     LI-LIST
               CA-CATLG/DB2
ITEM:     TB   FG-FILE GRP    PS-PSB      D2 (CA ONLY)
                              DL-DLI DBD  TB-SQL TBL   TJ-SQL JOIN
                              VS-VSAM     SQ-SEQ FILE  CQ-CICS QUE CJ-CICS JRNL

 NAME:         TELON.TRGEMPLZ_________ (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
 DESC:         EMPLOYEE TABLE CREATE EXAMPLE____________


 BASE:         TELON.TRGEMPLC_________ (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)
 RDBMS:        __________ (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

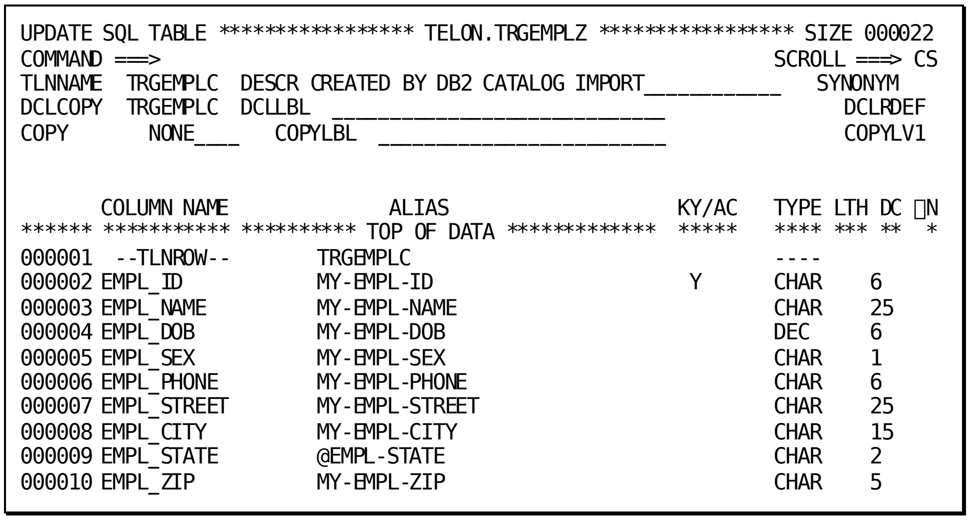
After you press Enter, the Update SQL Table screen appears.



```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000022
COMMAND ===>                                                    SCROLL ===> CS
TLNNAME  TRGEMPLC  DESCR  EMPLOYEE TABLE CREATE EXAMPLE_____     SYNONYM _ Y/N
DCLCOPY  TRGEMPLC  DCLLBL  _____________________________        DCLRDEF _ Y/N
COPY       NONE___    COPYLBL  _____________________________    COPYLV1 _ Y/N


       COLUMN NAME              ALIAS                    KY/AC    TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****    **** *** **  *
000001  --TLNROW--      TRGEMPLC                                  ----
000002 EMPL_ID          EMPL-ID                          Y        CHAR  6      Y
000003 EMPL_NAME        @EMPL-NAME                                CHAR  25     Y
000004 EMPL_DOB         @EMPL-DOB                                 DEC  6       Y
000005 EMPL_SEX         @EMPL-SEX                                 CHAR  1      Y
000006 EMPL_PHONE       @EMPL-PHONE                               CHAR  10     Y
000007 EMPL_STREET      @EMPL-STREET                              CHAR  25     Y
000008 EMPL_CITY        @EMPL-CITY                                CHAR  25     Y
000009 EMPL_STATE       @EMPL-STATE                               CHAR  2      Y
000010 EMPL_ZIP         @EMPL-ZIP                                 CHAR  5      Y
000011 EMPL_DOE         @EMPL-DOE                                 DEC   6      Y
000012 EMPL_DEPARTMENT  @EMPL-DEPARTMENT                          CHAR  3      Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                         DEC   5   2  N
000014 EMPL_HOURS       @EMPL-HOURS                               DEC   4   1  N
000015  --TLNROW--      TRGEMPVI                                  ----
000016 EMPL_ID          EMPL-ID                          Y        CHAR  6      Y
```

You can convert or ad a TLNROW in a DB2 table to define a temporary table that can be DGADDed into a program with the table to which it belongs. When a temporary table TLNROW if DGADDed to a program, the CA Telon Generator generates a DECLARE GLOBAL TEMPORARY TABLE command for the temporary table.

Identify the TLNROW as a temporary table in the Update SQL Table screen (D141) in the Data Administration by entering *TMP (for example, a row with a COLUMN NAME of --TLNROW--) TYPE field. The temporary table is made up of the columns belonging to this TLNROW.



```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000022
COMMAND ===>                                                    SCROLL ===> CS
TLNNAME   TRGEMPLC  DESCR CREATED BY DB2 CATALOG IMPORT___________    SYNONYM
DCLCOPY   TRGEMPLC  DCLLBL  ____________________________               DCLRDEF
COPY        NONE____   COPYLBL  ________________________               COPYLV1


       COLUMN NAME              ALIAS                    KY/AC   TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****   **** *** **  *
000001  --TLNROW--       TRGEMPLC                                ----
000002 EMPL_ID           MY-EMPL-ID                        Y     CHAR    6
000003 EMPL_NAME         MY-EMPL-NAME                            CHAR   25
000004 EMPL_DOB          MY-EMPL-DOB                             DEC     6
000005 EMPL_SEX          MY-EMPL-SEX                             CHAR    1
000006 EMPL_PHONE        MY-EMPL-PHONE                           CHAR    6
000007 EMPL_STREET       MY-EMPL-STREET                          CHAR   25
000008 EMPL_CITY         MY-EMPL-CITY                            CHAR   15
000009 EMPL_STATE        @EMPL-STATE                             CHAR    2
000010 EMPL_ZIP          MY-EMPL-ZIP                             CHAR    5
```

Updating SQL TABLE or TLNROW

Just as it is easy to create an SQL Table or TLNROW as shown in the last topic, it is also easy to update an existing SQL Table or TLNROW.

This topic shows how to:

- Update the table definition that was imported and used as a base in the previous topic (TELON.TRGEMPLC)
- Use this table definition in a data group for a particular screen definition
- Preview the generated code

Update the table definition on the Data Administration menu by typing **UP** in the FUNCTION field, **TB** in the ITEM field, and the SQL table name or TLNNAME in the NAME field (in this case **TELON.TRGEMPLC**).

```
DATA ADMINISTRATION MENU ************ ****************************************

COMMAND ==> _________________________________________________________________

FUNCTION:UP  CR-CREATE  UP-UPDATE       PU-PURGESH-SHOW  LI-LIST
                CA-CATLG/DB2
ITEM:     TB   FG-FILE GRP PS-PSB                D2 (CA ONLY)
                        DL-DLI DBD  TB-SQL TBL   TJ-SQL JOIN
                        VS-VSAM        SQ-SEQ FILE     CQ-CICS QUE
       CJ-CICS JRNL

NAME:     TELON.TRGEMPLZ____________   (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
DESC:     __________________________________________


BASE:     __________________________   (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)
RDBMS:   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

After you press Enter, you access the Update SQL Table screen.

```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000014
COMMAND ===>                                                  SCROLL ===> CSR
TLNNAME  TRGEMPLC  DESCR   CREATED BY DB2 CATALOG IMPORT_____  SYNONYM  _ Y/N
DCLCOPY  TRGEMPLC  DCLLBL  _______________________________     DCLRDEF  _ Y/N
COPY       NONE____   COPYLBL _______________________________  COPYLV1  _ Y/N


       COLUMN NAME            ALIAS                   KY/AC    TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****    **** *** **  *
000001  --TLNROW--      TRGEMPLC                               ----
000002 EMPL_ID          @EMPL-ID                                       CHAR  6
Y
000003 EMPL_NAME        @EMPL-NAME                             CHAR  25      Y
000004 EMPL_DOB         @EMPL-DOB                              DEC   6        Y
000005 EMPL_SEX         @EMPL-SEX                              CHAR  1       Y
000006 EMPL_PHONE       @EMPL-PHONE                            CHAR  10      Y
000007 EMPL_STREET      @EMPL-STREET                           CHAR  25      Y
000008 EMPL_CITY        @EMPL-CITY                             CHAR  25      Y
000009 EMPL_STATE       @EMPL-STATE                            CHAR  2       Y
000010 EMPL_ZIP         @EMPL-ZIP                              CHAR  5       Y
000011 EMPL_DOE         @EMPL-DOE                              DEC   6       Y
000012 EMPL_DEPARTMENT  @EMPL-DEPARTMENT                       CHAR  3       Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                      DEC   5   2   N
000014 EMPL_HOURS       @EMPL-HOURS                            DEC   4   1   N

****** *********** ******** BOTTOM OF DATA ************  *****    **** *** **  *
```

From the Update SQL Table screen, you can update the SQL table definition or TLNROW. The Update SQL Table screen shows one key specified with **Y**.

```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000014
COMMAND ==>                                                   SCROLL ==> CSR
TLNNAME   TRGEMPLC  DESCR   CREATED BY DB2 CATALOG IMPORT_____    SYNONYM  _ Y/N
DCLCOPY   TRGEMPLC  DCLLBL   ______________________________       DCLRDEF  _ Y/N
COPY       NONE____   COPYLBL ______________________________      COPYLV1  _ Y/N


       COLUMN NAME             ALIAS                   KY/AC    TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****    **** *** **  *
000001  --TLNROW--      TRGEMPLC                                ----
000002 EMPL_ID          @EMPL-ID                                Y       CHAR  6
Y
000003 EMPL_NAME        @EMPL-NAME                              CHAR  25      Y
000004 EMPL_DOB         @EMPL-DOB                               DEC   6        Y
000005 EMPL_SEX         @EMPL-SEX                               CHAR  1       Y
000006 EMPL_PHONE       @EMPL-PHONE                             CHAR  10      Y
000007 EMPL_STREET      @EMPL-STREET                            CHAR  25      Y
000008 EMPL_CITY        @EMPL-CITY                              CHAR  25      Y
000009 EMPL_STATE       @EMPL-STATE                             CHAR  2       Y
000010 EMPL_ZIP         @EMPL-ZIP                               CHAR  5       Y
000011 EMPL_DOE         @EMPL-DOE                               DEC   6       Y
000012 EMPL_DEPARTMENT  @EMPL-DEPARTMENT                        CHAR  3       Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                       DEC   5   2   N
000014 EMPL_HOURS       @EMPL-HOURS                             DEC   4   1   N

****** *********** ******** BOTTOM OF DATA ************  *****    **** *** **  *
```

If more than one key is specified with **Y**, the order of key reference is from top to bottom. For example, the Update SQL Table screen shows two Y's specified, the EMPL-ID is searched first (first item specified with Y) and the EMPL-NAME is searched next (second item specified with Y).

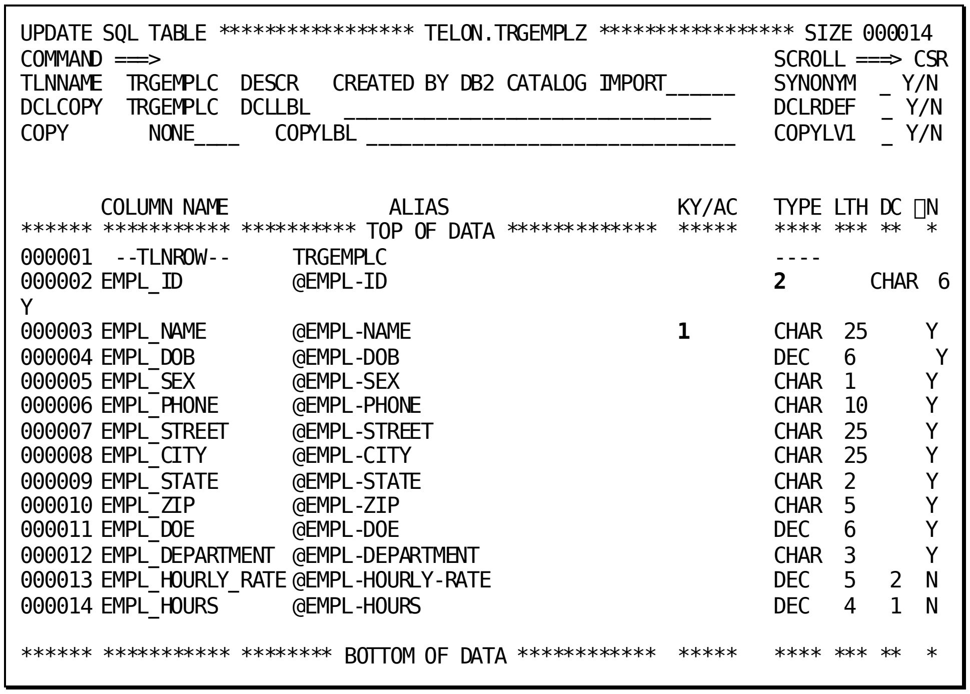
See the *Design Facility Reference Guide* for more detailed information on the parameters of this important screen.

```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000014
COMMAND ==>                                                   SCROLL ==> CSR
TLNNAME  TRGEMPLC  DESCR   CREATED BY DB2 CATALOG IMPORT_____     SYNONYM  _ Y/N
DCLCOPY  TRGEMPLC  DCLLBL   ______________________________      DCLRDEF  _ Y/N
COPY       NONE____   COPYLBL ______________________________    COPYLV1  _ Y/N


       COLUMN NAME             ALIAS                    KY/AC     TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****     **** *** **  *
000001  --TLNROW--      TRGEMPLC                                  ----
000002 EMPL_ID          @EMPL-ID                                  Y       CHAR  6
Y
000003 EMPL_NAME        @EMPL-NAME                       Y        CHAR  25     Y
000004 EMPL_DOB         @EMPL-DOB                                 DEC   6       Y
000005 EMPL_SEX         @EMPL-SEX                                 CHAR  1      Y
000006 EMPL_PHONE       @EMPL-PHONE                               CHAR  10     Y
000007 EMPL_STREET      @EMPL-STREET                              CHAR  25     Y
000008 EMPL_CITY        @EMPL-CITY                                CHAR  25     Y
000009 EMPL_STATE       @EMPL-STATE                               CHAR  2      Y
000010 EMPL_ZIP         @EMPL-ZIP                                 CHAR  5      Y
000011 EMPL_DOE         @EMPL-DOE                                 DEC   6      Y
000012 EMPL_DEPARTMENT  @EMPL-DEPARTMENT                          CHAR  3      Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                         DEC   5   2  N
000014 EMPL_HOURS       @EMPL-HOURS                               DEC   4   1  N

****** *********** ******** BOTTOM OF DATA ************  *****     **** *** **  *
```

Another way to specify the order of the keys is by using numerals instead of Ys.

The next examples, however, show a different ordering of the keys. The **1** specified for EMPL-NAME means that it is the first key searched. The **2** specified for EMPL-ID means that it is searched next on the database.

```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000014
COMMAND ===>                                                    SCROLL ===> CSR
TLNNAME  TRGEMPLC  DESCR   CREATED BY DB2 CATALOG IMPORT_____   SYNONYM  _ Y/N
DCLCOPY  TRGEMPLC  DCLLBL   _______________________________     DCLRDEF  _ Y/N
COPY       NONE___   COPYLBL _______________________________    COPYLV1  _ Y/N


       COLUMN NAME              ALIAS                  KY/AC    TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****    **** *** **  *
000001  --TLNROW--      TRGEMPLC                                ----
000002 EMPL_ID          @EMPL-ID                                1       CHAR  6
Y
000003 EMPL_NAME        @EMPL-NAME                     2        CHAR  25      Y
000004 EMPL_DOB         @EMPL-DOB                               DEC   6        Y
000005 EMPL_SEX         @EMPL-SEX                               CHAR  1       Y
000006 EMPL_PHONE       @EMPL-PHONE                             CHAR  10      Y
000007 EMPL_STREET      @EMPL-STREET                            CHAR  25      Y
000008 EMPL_CITY        @EMPL-CITY                              CHAR  25      Y
000009 EMPL_STATE       @EMPL-STATE                             CHAR  2       Y
000010 EMPL_ZIP         @EMPL-ZIP                               CHAR  5       Y
000011 EMPL_DOE         @EMPL-DOE                               DEC   6       Y
000012 EMPL_DEPARTMENT  @EMPL-DEPARTMENT                        CHAR  3       Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                       DEC   5   2   N
000014 EMPL_HOURS       @EMPL-HOURS                             DEC   4   1   N

****** *********** ******** BOTTOM OF DATA ************  *****    **** *** **  *
```



```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000014
COMMAND ==>                                                     SCROLL ==> CSR
TLNNAME  TRGEMPLC  DESCR   CREATED BY DB2 CATALOG IMPORT______  SYNONYM  _ Y/N
DCLCOPY  TRGEMPLC  DCLLBL  ______________________________       DCLRDEF  _ Y/N
COPY       NONE____  COPYLBL ______________________________     COPYLV1  _ Y/N


       COLUMN NAME             ALIAS                    KY/AC   TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****   **** *** **  *
000001  --TLNROW--     TRGEMPLC                                 ----
000002 EMPL_ID         @EMPL-ID                                 2      CHAR  6
Y
000003 EMPL_NAME       @EMPL-NAME                        1      CHAR  25      Y
000004 EMPL_DOB        @EMPL-DOB                                DEC   6        Y
000005 EMPL_SEX        @EMPL-SEX                                CHAR  1       Y
000006 EMPL_PHONE      @EMPL-PHONE                              CHAR  10      Y
000007 EMPL_STREET     @EMPL-STREET                             CHAR  25      Y
000008 EMPL_CITY       @EMPL-CITY                               CHAR  25      Y
000009 EMPL_STATE      @EMPL-STATE                              CHAR  2       Y
000010 EMPL_ZIP        @EMPL-ZIP                                CHAR  5       Y
000011 EMPL_DOE        @EMPL-DOE                                DEC   6       Y
000012 EMPL_DEPARTMENT @EMPL-DEPARTMENT                         CHAR  3       Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                       DEC   5   2   N
000014 EMPL_HOURS      @EMPL-HOURS                              DEC   4   1   N

****** *********** ******** BOTTOM OF DATA ************  *****   **** *** **  *
```

The next three screens show how you can create new TLNROWs by copying existing TLNROWs using ISPF-like line commands.

The first screen sample shows the copying of a block of text (**CC CC**) to place it after a specified line (**A**).

```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000014
COMMAND ===>                                                  SCROLL ===> CSR
TLNNAME   TRGEMPLC  DESCR   CREATED BY DB2 CATALOG IMPORT_____    SYNONYM  _ Y/N
DCLCOPY   TRGEMPLC  DCLLBL   ______________________________    DCLRDEF  _ Y/N
COPY        NONE____   COPYLBL ______________________________    COPYLV1  _ Y/N


       COLUMN NAME              ALIAS                    KY/AC    TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****    **** *** **  *
CC 001  --TLNROW--      TRGEMPLC                                  ----
000002 EMPL_ID          @EMPL-ID                                  Y       CHAR  6
Y
000003 EMPL_NAME        @EMPL-NAME                                CHAR  25     Y
000004 EMPL_DOB         @EMPL-DOB                                 DEC   6       Y
000005 EMPL_SEX         @EMPL-SEX                                 CHAR  1      Y
000006 EMPL_PHONE       @EMPL-PHONE                               CHAR  10     Y
000007 EMPL_STREET      @EMPL-STREET                              CHAR  25     Y
CC 008 EMPL_CITY        @EMPL-CITY                                CHAR  25     Y
000009 EMPL_STATE       @EMPL-STATE                               CHAR  2      Y
000010 EMPL_ZIP         @EMPL-ZIP                                 CHAR  5      Y
000011 EMPL_DOE         @EMPL-DOE                                 DEC   6      Y
000012 EMPL_DEPARTMENT  @EMPL-DEPARTMENT                          CHAR  3      Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                         DEC   5   2  N
A 0014 EMPL_HOURS       @EMPL-HOURS                               DEC   4   1  N
000015 --TLNROW--       TRGEMPLC                                  ----
___

****** *********** ******** BOTTOM OF DATA ************  *****    **** *** **  *
```

The second screen demonstrates the changing of the name of the TLNROW from TRGEMPLC to **TRGEMPVI**, by typing over "TRGEMPLC."

```
UPDATE SQL TABLE ***************** TELON.TRGEMPLZ ***************** SIZE 000014
COMMAND ===>                                                  SCROLL ===> CSR
TLNNAME  TRGEMPLC  DESCR   CREATED BY DB2 CATALOG IMPORT_____  SYNONYM  _ Y/N
DCLCOPY  TRGEMPLC  DCLLBL   ______________________________     DCLRDEF  _ Y/N
COPY       NONE___    COPYLBL ______________________________   COPYLV1  _ Y/N



       COLUMN NAME              ALIAS                    KY/AC    TYPE LTH DC □N
****** *********** ********** TOP OF DATA *************  *****    **** *** **  *
000001  --TLNROW--      TRGEMPLC                                  ----
000002 EMPL_ID          @EMPL-ID                                  Y       CHAR  6
Y
000003 EMPL_NAME        @EMPL-NAME                                CHAR  25      Y
000004 EMPL_DOB         @EMPL-DOB                                 DEC   6        Y
000005 EMPL_SEX         @EMPL-SEX                                 CHAR  1       Y
000006 EMPL_PHONE       @EMPL-PHONE                               CHAR  10      Y
000007 EMPL_STREET      @EMPL-STREET                              CHAR  25      Y
000008 EMPL_CITY        @EMPL-CITY                                CHAR  25      Y
000009 EMPL_STATE       @EMPL-STATE                               CHAR  2       Y
000010 EMPL_ZIP         @EMPL-ZIP                                 CHAR  5       Y
000011 EMPL_DOE         @EMPL-DOE                                 DEC   6       Y
000012 EMPL_DEPARTMENT  @EMPL-DEPARTMENT                          CHAR  3       Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                         DEC   5   2   N
000014 EMPL_HOURS       @EMPL-HOURS                               DEC   4   1   N
000015 --TLNROW--       TRGEMPVI                                  ----
000016 EMPL_ID          @EMPL-ID                                  Y       CHAR  6
Y
```

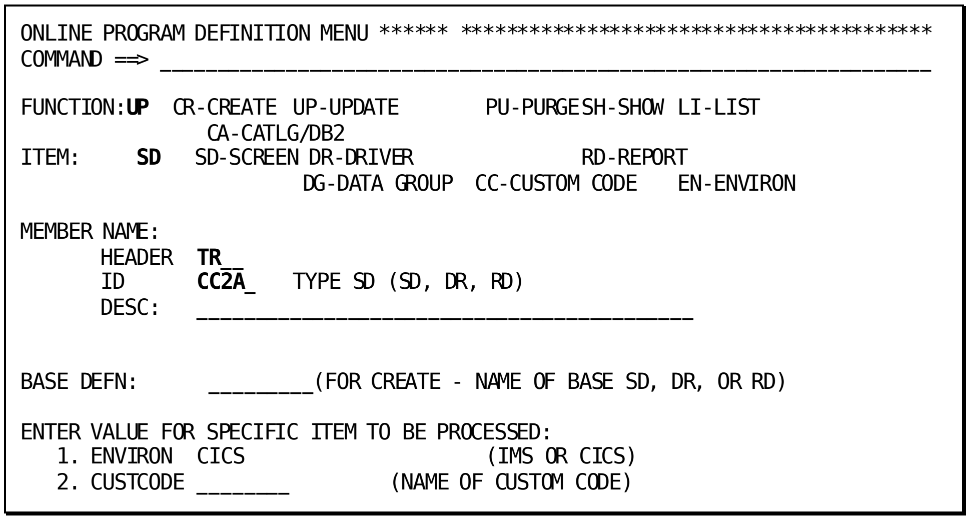
The third screen shows the paging down (PF8) of the information shown.

```
UPDATE SQL TABLE **************** TELON.TRGEMPLZ **************** SIZE 000014
COMMAND ===>                                                     SCROLL ===> CSR
TLNNAME  TRGEMPLC  DESCR   CREATED BY DB2 CATALOG IMPORT_____    SYNONYM  _ Y/N
DCLCOPY  TRGEMPLC  DCLLBL   ______________________________       DCLRDEF  _ Y/N
COPY       NONE___    COPYLBL ______________________________     COPYLV1  _ Y/N



       COLUMN NAME             ALIAS                    KY/AC    TYPE LTH DC □N
****** *********** ********** TOP OF DATA ************  *****    **** *** **  *
000008 EMPL_CITY        @EMPL-CITY                               CHAR  25     Y
000009 EMPL_STATE       @EMPL-STATE                              CHAR  2      Y
000010 EMPL_ZIP         @EMPL-ZIP                                CHAR  5      Y
000011 EMPL_DOE         @EMPL-DOE                                DEC   6      Y
000012 EMPL_DEPARTMENT  @EMPL-DEPARTMENT                         CHAR  3      Y
000013 EMPL_HOURLY_RATE @EMPL-HOURLY-RATE                        DEC   5   2  N
000014 EMPL_HOURS       @EMPL-HOURS                              DEC   4   1  N
000015 --TLNROW--       TRGEMPVI                                 ----
000016 EMPL_ID          @EMPL-ID                                 Y       CHAR  6
Y
000017 EMPL_NAME        @EMPL-NAME                               CHAR  25     Y
000018 EMPL_DOB         @EMPL-DOB                                DEC   6       Y
000019 EMPL_SEX         @EMPL-SEX                                CHAR  1      Y
000020 EMPL_PHONE       @EMPL-PHONE                              CHAR  10     Y
000021 EMPL_STREET      @EMPL-STREET                             CHAR  25     Y
000022 EMPL_CITY        @EMPL-CITY                               CHAR  25     Y
****** *********** ******** BOTTOM OF DATA ************  *****   **** *** **  *
```

You use TLNROWs to create different "views" of the table in CA Telon. This is particularly useful for programmers when they create programs that have different requirements for accessing a table. There is no limit to the number of TLNROWs that can be created for a table. Refer to the *Design Facility Reference Guide* for the specific line commands you can use. The next section presents an example of how TLNROWs can be used in a data group.

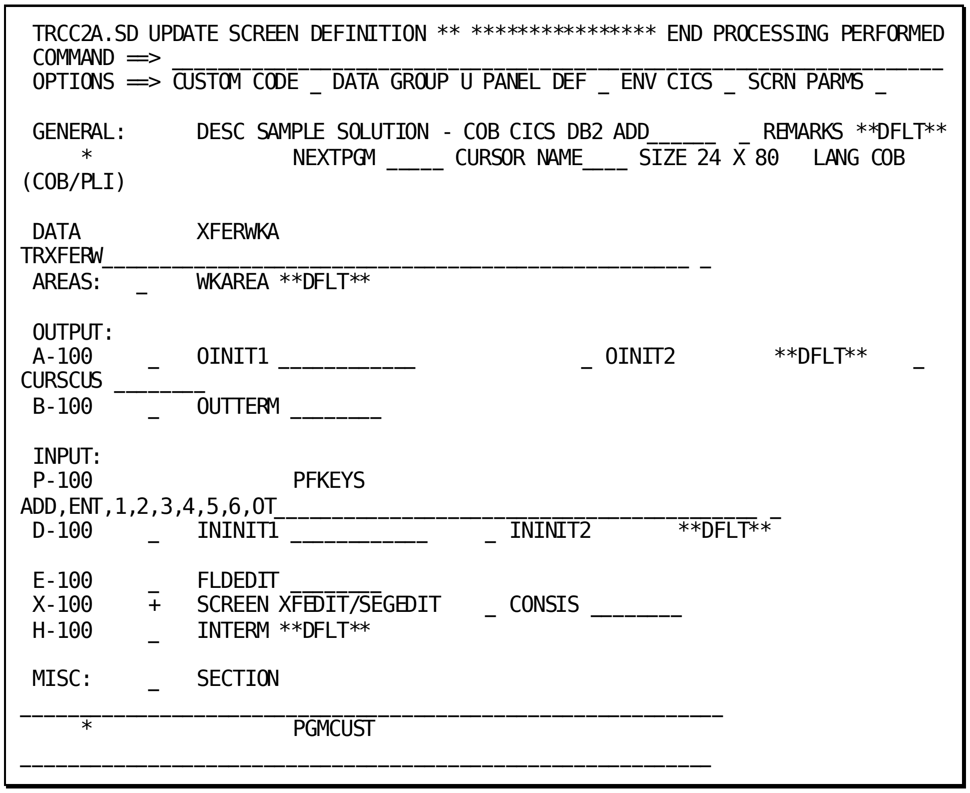
When you type **=4** on the COMMAND line on the screen and press Enter, CA Telon displays the Online Program Definition menu shown in the Online Program Definition Menu screen.

From this screen, type **UP** in the FUNCTION field, **SD** in the ITEM field and the appropriate header and ID in the HEADER and ID fields (in this case, **TR** and **CC2A**). press Enter to display the Update Screen Definition screen.



```
ONLINE PROGRAM DEFINITION MENU ****** ****************************************
COMMAND ==> __________________________________________________________________

FUNCTION:UP  CR-CREATE  UP-UPDATE       PU-PURGESH-SHOW  LI-LIST
                CA-CATLG/DB2
ITEM:     SD   SD-SCREEN DR-DRIVER              RD-REPORT
                        DG-DATA GROUP  CC-CUSTOM CODE    EN-ENVIRON

MEMBER NAME:
       HEADER  TR__
       ID      CC2A_   TYPE SD (SD, DR, RD)
       DESC:   ________________________________________


BASE DEFN:      ________(FOR CREATE - NAME OF BASE SD, DR, OR RD)

ENTER VALUE FOR SPECIFIC ITEM TO BE PROCESSED:
   1. ENVIRON  CICS                    (IMS OR CICS)
   2. CUSTCODE _______         (NAME OF CUSTOM CODE)
```

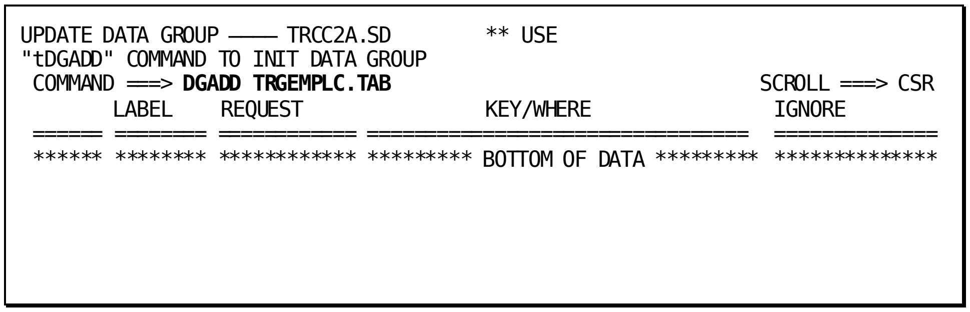
Type a non-blank character (in this case, **U**) in the DATA GROUP field and press Enter to display the Update Data Group screen.



```
 TRCC2A.SD UPDATE SCREEN DEFINITION ** **************** END PROCESSING PERFORMED
 COMMAND ==> _________________________________________________________________
 OPTIONS ==> CUSTOM CODE _ DATA GROUP U PANEL DEF _ ENV CICS _ SCRN PARMS _

 GENERAL:      DESC SAMPLE SOLUTION - COB CICS DB2 ADD_____  _ REMARKS **DFLT**
     *                 NEXTPGM ____ CURSOR NAME___ SIZE 24 X 80   LANG COB
(COB/PLI)

 DATA          XFERWKA
TRXFERW_______________________________________________ _
 AREAS:    _     WKAREA **DFLT**

 OUTPUT:
 A-100      _    OINIT1 ___________             _ OINIT2        **DFLT**     _
CURSCUS _______
 B-100      _    OUTTERM _______

 INPUT:
 P-100                  PFKEYS
ADD,ENT,1,2,3,4,5,6,OT_______________________________________ _
 D-100      _    ININIT1 ___________     _ ININIT2       **DFLT**

 E-100      _    FLDEDIT _______
 X-100      +    SCREEN XFEDIT/SEGEDIT    _ CONSIS _______
 H-100      _    INTERM **DFLT**

 MISC:      _    SECTION
_____________________________________________________________
     *                  PGMCUST
____________________________________________________________
```

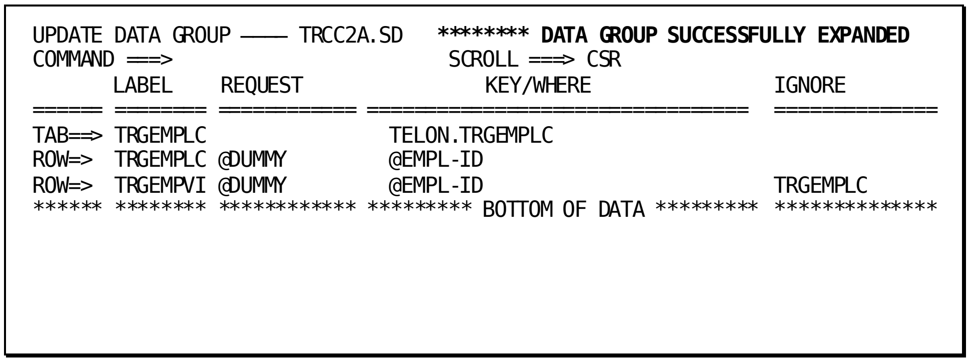
On the COMMAND line of the Update Data Group screen, type the appropriate DGADD (Data Group Add) command. The DGADD command specifies the TLNNAME of the SQL object in option 2 needed by this screen definition, qualified by what type of object it is.

TB indicates a table and TJ indicates table join criteria. This example uses the table that has just been updated in option 2. Notice that the table has two TLNROWs defined for it. For a full list of DGADD commands, refer to the *Design Facility Reference Guide* (Update Data Group screen).



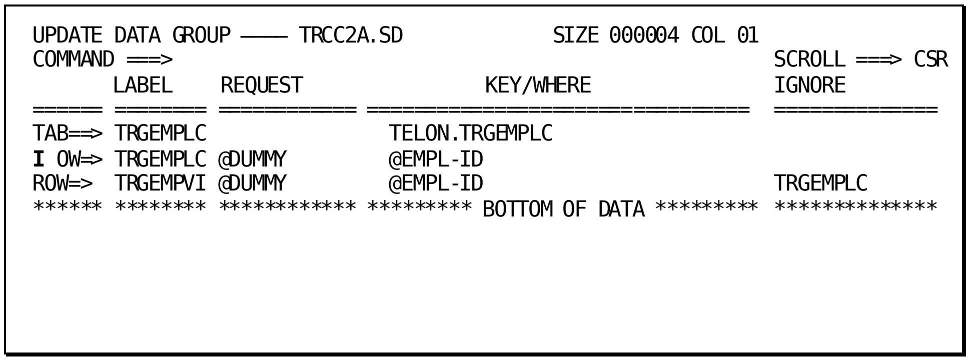
```
UPDATE DATA GROUP ---- TRCC2A.SD        ** USE
"tDGADD" COMMAND TO INIT DATA GROUP
 COMMAND ===> DGADD TRGEMPLC.TAB                                SCROLL ===> CSR
        LABEL    REQUEST              KEY/WHERE                  IGNORE
 ====== ======== ============ =================================  ==============
 ****** ******** ************ ********* BOTTOM OF DATA *********  **************
```

After you press Enter, CA Telon initializes the Update Data Group screen with the data group information requested. Notice how CA Telon displays the message **DATA GROUP SUCCESSFULLY EXPANDED** in the upper-right corner.



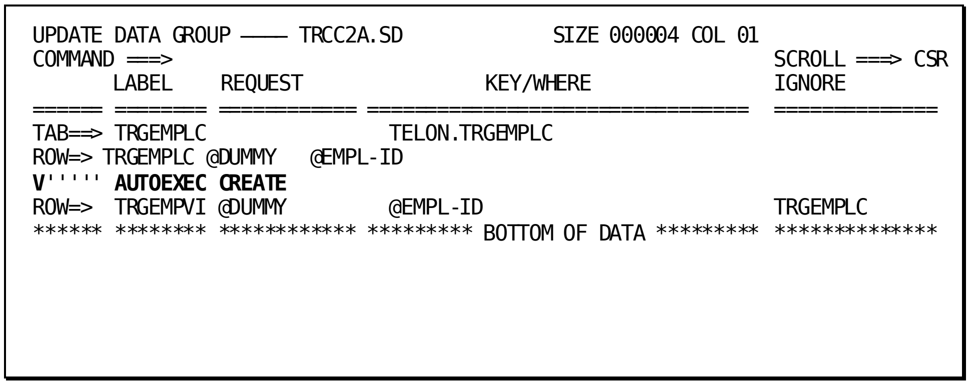
```
 UPDATE DATA GROUP ---- TRCC2A.SD   ******** DATA GROUP SUCCESSFULLY EXPANDED
 COMMAND ===>                        SCROLL ===> CSR
        LABEL    REQUEST              KEY/WHERE                  IGNORE
 ====== ======== ============ =================================  ==============
 TAB==> TRGEMPLC                TELON.TRGEMPLC
 ROW=>  TRGEMPLC @DUMMY         @EMPL-ID
 ROW=>  TRGEMPVI @DUMMY         @EMPL-ID                         TRGEMPLC
 ****** ******** ************ ********* BOTTOM OF DATA *********  **************
```

Using the CA Telon editor, insert a line between the rows (I line command) on the Update Data Group screen.



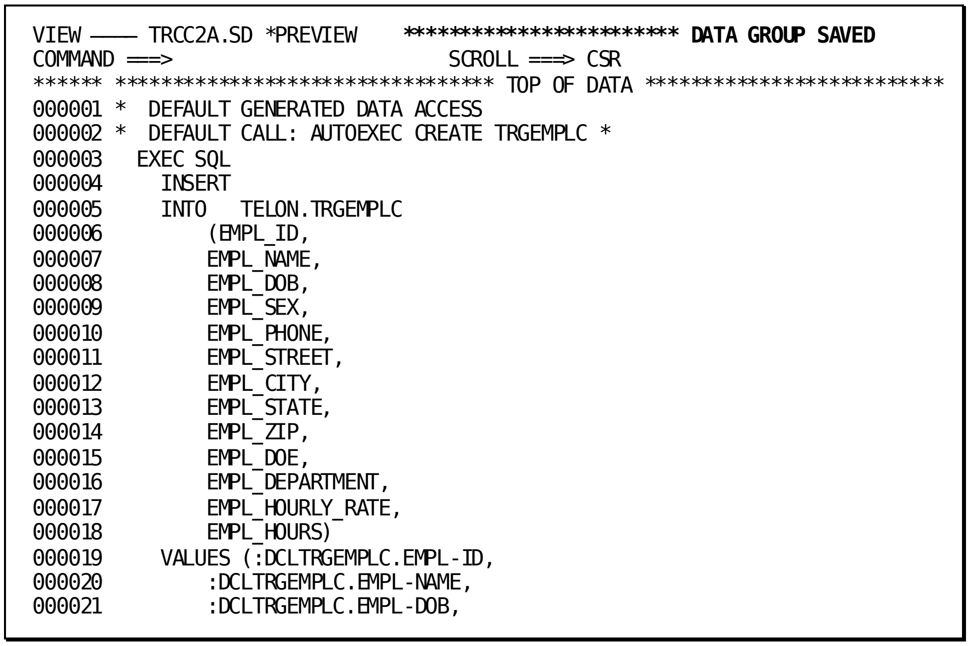
```
UPDATE DATA GROUP ---- TRCC2A.SD            SIZE 000004 COL 01
COMMAND ===>                                                       SCROLL ===> CSR
       LABEL    REQUEST                KEY/WHERE                   IGNORE
====== ======== ============ ================================ ==============
TAB==> TRGEMPLC              TELON.TRGEMPLC
I OW=> TRGEMPLC @DUMMY       @EMPL-ID
ROW=>  TRGEMPVI @DUMMY       @EMPL-ID                         TRGEMPLC
****** ******** ************ ********* BOTTOM OF DATA ********* **************
```

Then type **V** in the line command, **AUTOEXEC** as the LABEL, and **CREATE** as the REQUEST.



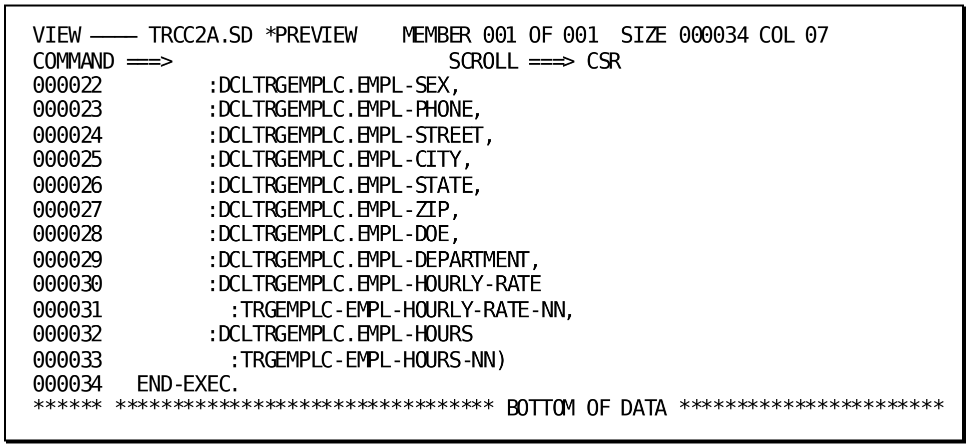
```
UPDATE DATA GROUP ---- TRCC2A.SD            SIZE 000004 COL 01
COMMAND ===>                                                       SCROLL ===> CSR
       LABEL    REQUEST                KEY/WHERE                   IGNORE
====== ======== ============ ================================ ==============
TAB==> TRGEMPLC              TELON.TRGEMPLC
ROW=> TRGEMPLC @DUMMY   @EMPL-ID
V''''' AUTOEXEC CREATE
ROW=>  TRGEMPVI @DUMMY       @EMPL-ID                         TRGEMPLC
****** ******** ************ ********* BOTTOM OF DATA ********* **************
```

After you press Enter, the SQL Preview screen appears. This screen enables you to view online what the generated code will look like for this request on TLNROW TRGEMPLC.



```
VIEW ---- TRCC2A.SD *PREVIEW    ************************ DATA GROUP SAVED
COMMAND ===>                               SCROLL ===> CSR
****** ********************************* TOP OF DATA *************************
000001 *  DEFAULT GENERATED DATA ACCESS
000002 *  DEFAULT CALL: AUTOEXEC CREATE TRGEMPLC *
000003   EXEC SQL
000004     INSERT
000005     INTO   TELON.TRGEMPLC
000006         (EMPL_ID,
000007         EMPL_NAME,
000008         EMPL_DOB,
000009         EMPL_SEX,
000010         EMPL_PHONE,
000011         EMPL_STREET,
000012         EMPL_CITY,
000013         EMPL_STATE,
000014         EMPL_ZIP,
000015         EMPL_DOE,
000016         EMPL_DEPARTMENT,
000017         EMPL_HOURLY_RATE,
000018         EMPL_HOURS)
000019     VALUES (:DCLTRGEMPLC.EMPL-ID,
000020         :DCLTRGEMPLC.EMPL-NAME,
000021         :DCLTRGEMPLC.EMPL-DOB,
```

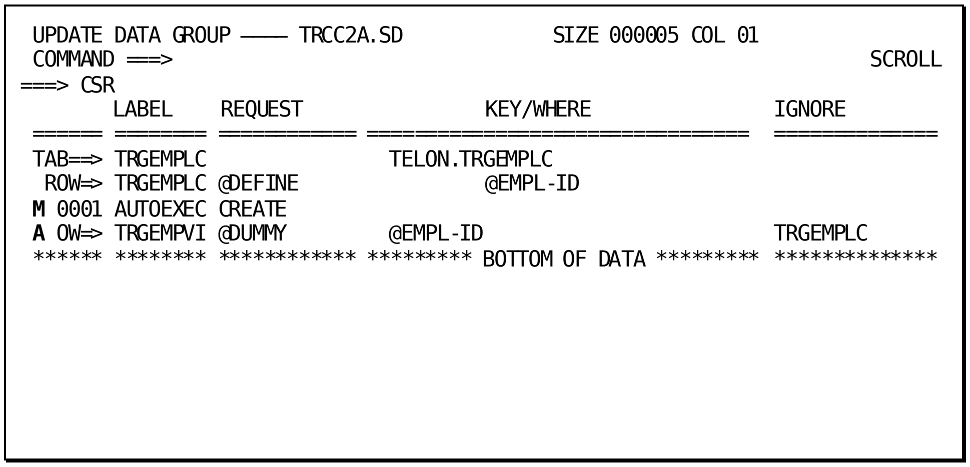
When you press PF8, you can see the rest of the generated code.



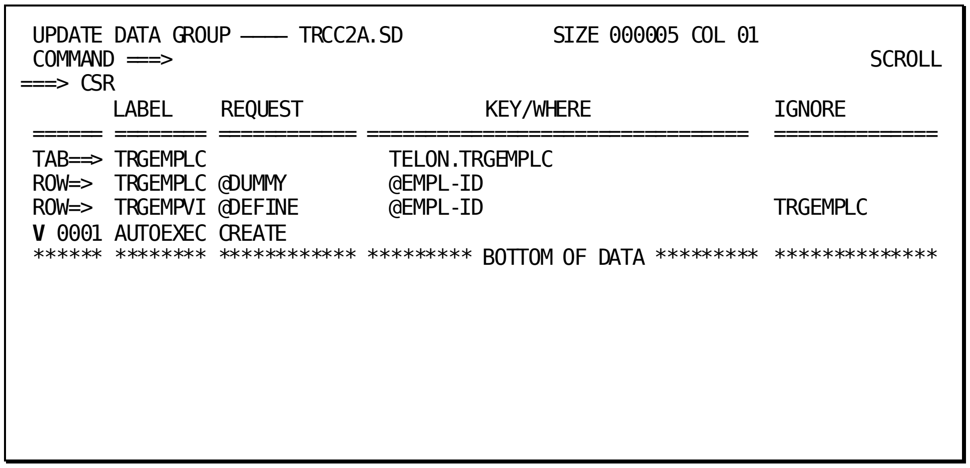
```
VIEW ---- TRCC2A.SD *PREVIEW    MEMBER 001 OF 001  SIZE 000034 COL 07
COMMAND ===>                               SCROLL ===> CSR
000022         :DCLTRGEMPLC.EMPL-SEX,
000023         :DCLTRGEMPLC.EMPL-PHONE,
000024         :DCLTRGEMPLC.EMPL-STREET,
000025         :DCLTRGEMPLC.EMPL-CITY,
000026         :DCLTRGEMPLC.EMPL-STATE,
000027         :DCLTRGEMPLC.EMPL-ZIP,
000028         :DCLTRGEMPLC.EMPL-DOE,
000029         :DCLTRGEMPLC.EMPL-DEPARTMENT,
000030         :DCLTRGEMPLC.EMPL-HOURLY-RATE
000031           :TRGEMPLC-EMPL-HOURLY-RATE-NN,
000032         :DCLTRGEMPLC.EMPL-HOURS
000033           :TRGEMPLC-EMPL-HOURS-NN)
000034   END-EXEC.
****** ********************************* BOTTOM OF DATA **********************
```

After you press PF3, you return to the Update Data Group screen. Move the AUTOEXEC command after the TRGEMPVI TLNROW, using the Move (M) and After (A) commands as shown.

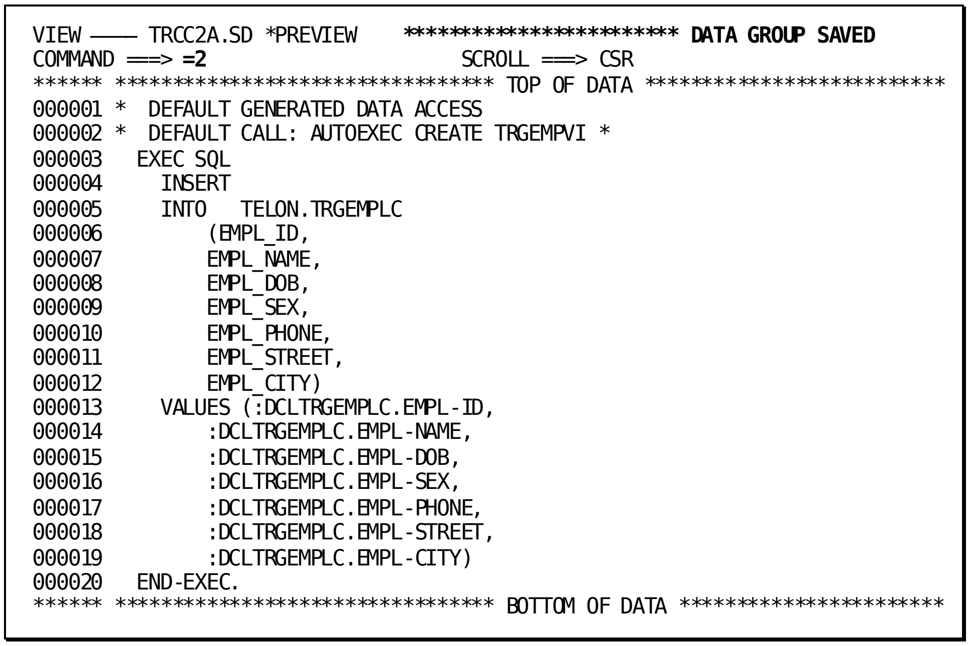
Press Enter and the AUTOEXEC line moves as shown.


```
 UPDATE DATA GROUP ——— TRCC2A.SD           SIZE 000005 COL 01
 COMMAND ===>                                                      SCROLL
===> CSR
       LABEL    REQUEST               KEY/WHERE                IGNORE
 ===== ======= =========== ============================== =============
 TAB==> TRGEMPLC             TELON.TRGEMPLC
  ROW=> TRGEMPLC @DEFINE             @EMPL-ID
 M 0001 AUTOEXEC CREATE
 A OW=> TRGEMPVI @DUMMY      @EMPL-ID                      TRGEMPLC
 ****** ******** ************ ********* BOTTOM OF DATA ********* **************
```

Type **V** in the left-hand column of the AUTOEXEC line and press Enter. CA Telon displays the SQL Preview screen and shows the generated code for this request. This is an example of a way multiple TLNROWs can be used.


```
 UPDATE DATA GROUP ——— TRCC2A.SD           SIZE 000005 COL 01
 COMMAND ===>                                                      SCROLL
===> CSR
       LABEL    REQUEST               KEY/WHERE                IGNORE
 ===== ======= =========== ============================== =============
 TAB==> TRGEMPLC             TELON.TRGEMPLC
 ROW=>  TRGEMPLC @DUMMY      @EMPL-ID
 ROW=>  TRGEMPVI @DEFINE     @EMPL-ID                      TRGEMPLC
 V 0001 AUTOEXEC CREATE
 ****** ******** ************ ********* BOTTOM OF DATA ********* **************
```

Type **=2** in the COMMAND field to return to the Data Administration menu.

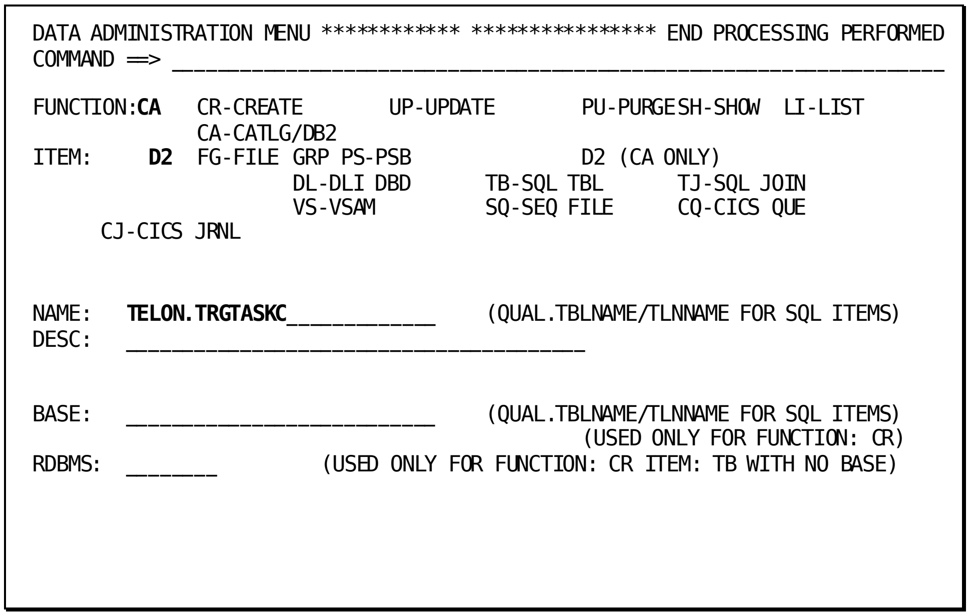


```
VIEW ---- TRCC2A.SD *PREVIEW    *********************** DATA GROUP SAVED
COMMAND ===> =2                          SCROLL ===> CSR
****** ******************************** TOP OF DATA *************************
000001 *  DEFAULT GENERATED DATA ACCESS
000002 *  DEFAULT CALL: AUTOEXEC CREATE TRGEMPVI *
000003   EXEC SQL
000004     INSERT
000005     INTO   TELON.TRGEMPLC
000006         (EMPL_ID,
000007         EMPL_NAME,
000008         EMPL_DOB,
000009         EMPL_SEX,
000010         EMPL_PHONE,
000011         EMPL_STREET,
000012         EMPL_CITY)
000013     VALUES (:DCLTRGEMPLC.EMPL-ID,
000014         :DCLTRGEMPLC.EMPL-NAME,
000015         :DCLTRGEMPLC.EMPL-DOB,
000016         :DCLTRGEMPLC.EMPL-SEX,
000017         :DCLTRGEMPLC.EMPL-PHONE,
000018         :DCLTRGEMPLC.EMPL-STREET,
000019         :DCLTRGEMPLC.EMPL-CITY)
000020   END-EXEC.
****** ******************************** BOTTOM OF DATA **********************
```

Creating TABLE JOIN Criteria

An important SQL task is joining together two or more SQL tables so that all of these tables can be accessed with one command (rather than several separate commands).

This topic shows how to create the criteria to join three tables: TELON.TRGEMPLC, TELON.TRGTASKC, and TELON.TRGTIMEC. To do this, we must first import two or more tables into CA Telon. From the Data Administration menu, type **CA** in the FUNCTION field, **D2** or **TB** in the ITEM field, and the table name in the NAME field (in this case, **TELON.TRGTASKC**) and press Enter.



```
DATA ADMINISTRATION MENU ************ **************** END PROCESSING PERFORMED
COMMAND ==> _______________________________________________________________

FUNCTION:CA   CR-CREATE       UP-UPDATE       PU-PURGESH-SHOW  LI-LIST
              CA-CATLG/DB2
ITEM:     D2  FG-FILE GRP PS-PSB              D2 (CA ONLY)
                      DL-DLI DBD      TB-SQL TBL      TJ-SQL JOIN
                      VS-VSAM         SQ-SEQ FILE     CQ-CICS QUE
      CJ-CICS JRNL


NAME:   TELON.TRGTASKC____________    (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
DESC:   ________________________________________


BASE:   __________________________    (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                              (USED ONLY FOR FUNCTION: CR)
RDBMS:  ________         (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Catalog/Import DB2 Tables screen. Type **I** next to the table names you want to import (in this case, TELON.TRGTASKC and TELON.TRGTIMEC).

```
CATALOG/IMPORT DB2 TABLES *********** ****************************************
 COMMAND ==>                                                  PAGE 01 MORE

 SELECT THE DB2 TABLES TO BE IMPORTED INTO THE TDF
  **QUAL** ***TABLENAME*** COLS   TLN   TLNNAME         USERID  UPDATE
 I TELON  TRGTASKC      9       N
  TELON  TRGTASKD      9       N
  TELON  TRGTASKE      9       N
  TELON  TRGTASKF      9       N
  TELON  TRGTASK0      9       N
  TELON  TRGTASK1      9       Y      TRGTASK1MOSSM   07/11/2001
  TELON  TRGTASK2      9       Y      TRGTASK2        ADAMS   04/05/2000
  TELON  TRGTASK3      9       N
  TELON  TRGTASK4      9       N
  TELON  TRGTASK5      9       N
  TELON  TRGTASK6      9       N
  TELON  TRGTASK7      9       N
  TELON  TRGTASK8      9       N
  TELON  TRGTASK9      9       N
  TELON  TRGTIME       10      N
  TELON  TRGTIMEA      10      N
  TELON  TRGTIMEB      10      N
I TELON  TRGTIMEC      10      N
  TELON  TRGTIMED      10      N
```

Press Enter and the screen refreshes and displays the message **IMPORT OK** to show that the IMPORT was successful.

```
CATALOG/IMPORT DB2 TABLES *********** ****************************************
 COMMAND ==>                                                  PAGE 01 MORE

 SELECT THE DB2 TABLES TO BE IMPORTED INTO THE TDF
  **QUAL** ***TABLENAME*** COLS   TLN   TLNNAME         USERID  UPDATE
  TELON  TRGTASKC      9       Y      TRGTASKC                * IMPORT OK
  TELON  TRGTASKD      9       N
  TELON  TRGTASKE      9       N
  TELON  TRGTASKF      9       N
  TELON  TRGTASK0      9       N
  TELON  TRGTASK1      9       Y      TRGTASK1MOSSM   07/11/2001
  TELON  TRGTASK2      9       Y      TRGTASK2        ADAMS   04/05/2000
  TELON  TRGTASK3      9       N
  TELON  TRGTASK4      9       N
  TELON  TRGTASK5      9       N
  TELON  TRGTASK6      9       N
  TELON  TRGTASK7      9       N
  TELON  TRGTASK8      9       N
  TELON  TRGTASK9      9       N
  TELON  TRGTIME       10      N
  TELON  TRGTIMEA      10      N
  TELON  TRGTIMEB      10      N
  TELON  TRGTIMEC      10      Y      TRGTIMEC                * IMPORT OK
  TELON  TRGTIMED      10      N
```

Press PF3 to return to the Data Administration menu. From the Data Administration menu, type **UP** in the FUNCTION field, **TB** in the ITEM field, and the table name in the NAME field (in this case, **TELON.TRGTASKC**) and press Enter.

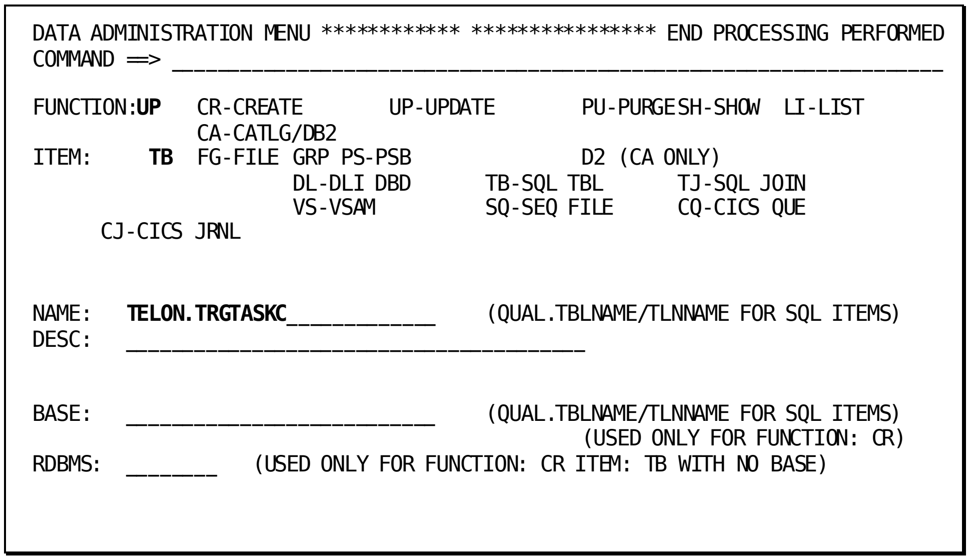
```
DATA ADMINISTRATION MENU ************ **************** END PROCESSING PERFORMED
COMMAND ==> ___________________________________________________________________

FUNCTION: UP  CR-CREATE       UP-UPDATE  PU-PURGE   SH-SHOW   LI-LIST
                CA-CATLG/DB2
ITEM:     TB  FG-FILE GRP     PS-PSB          D2 (CA ONLY)
                DL-DLI DBD      TB-SQL TBL      TJ-SQL JOIN
                VS-VSAM         SQ-SEQ FILE     CQ-CICS QUE
      CJ-CICS JRNL

NAME:   TELON.TRGTASKC____________    (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
DESC:   ________________________________________


BASE:   __________________________    (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                              (USED ONLY FOR FUNCTION: CR)
RDBMS:  ________         (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update SQL Table screen. This screen shows all the task-related TLNROW data. You could make changes here, if necessary.

```
UPDATE SQL TABLE ******************** TELON.TRGTASKC ************ SIZE 000010
COMMAND ==>                                                   SCROLL ===>
CSR
TLNNAME TRGTASKC DESCR  CREATED BY DB2 CATALOG IMPORT__________   SYNONYM _ Y/N
DCLCOPY TRGTASKC DCLLBL ______________________________
       DCLRDEF _ Y/N
COPY  NONE____ COPYLBL ______________________________
       COPYLV1 _ Y/N

     COLUMN NAME           ALIAS                KY/AC TYPE LTH   DC ^N
****** ****************** ******** TOP OF DATA ********* ** *  **** ***** ** *
000001 --TLNROW--       TRGTASKC                 ----
000002 TASK_EMPL_ID     @TASK-EMPL-ID            CHAR     6      Y
000003 TASK_PROJ_ID     @TASK-PROJ-ID            CHAR     4      Y
000004 TASK_TASK_ID     @TASK-TASK-ID            CHAR     4      Y
000005 TASK_P_DESC      @TASK-P-DESC             CHAR    22      Y
000006 TASK_P_PRTY      @TASK-P-PRTY             CHAR     2      Y
000007 TASK_P_CODE      @TASK-P-CODE             CHAR     6      Y
000008 TASK_T_DESC      @TASK-T-DESC             CHAR    22      Y
000009 TASK_T_PRTY      @TASK-T-PRTY             CHAR     2      Y
000010 TASK_T_CODE      @TASK-T-CODE             CHAR     6      Y
****** ****************** ******* BOTTOM OF DATA ******* ** *  **** ***** ** *
```

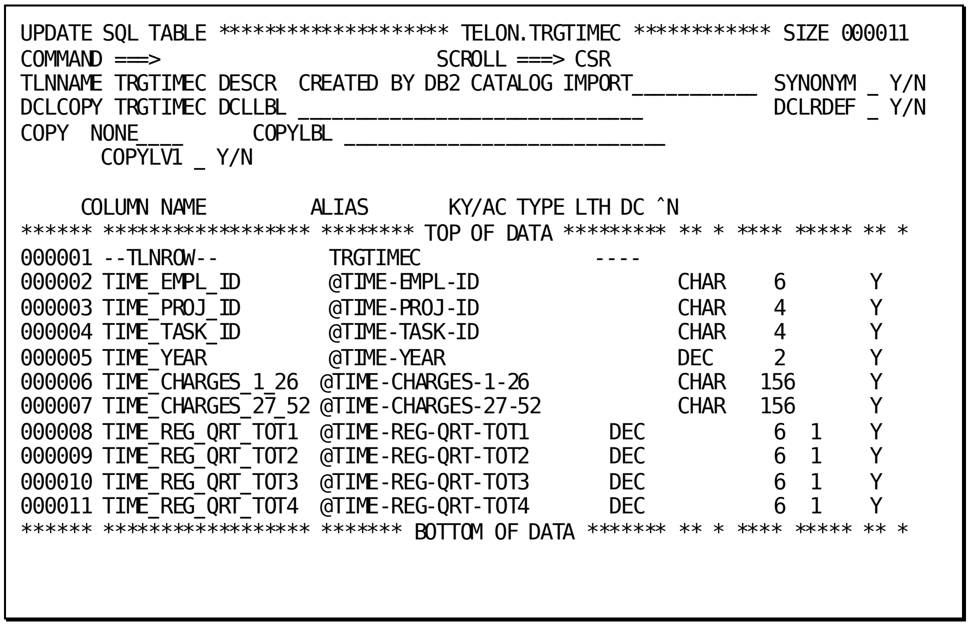
Press PF3 to return to the Data Administration menu.

From the Data Administration menu, type **UP** in the FUNCTION field, **TB** in the ITEM field, and the table name in the NAME field (in this case, **TELON.TRGTIMEC**) and press Enter.



```
DATA ADMINISTRATION MENU ************ **************** END PROCESSING PERFORMED
COMMAND ==> ___________________________________________________________________

FUNCTION:UP   CR-CREATE       UP-UPDATE       PU-PURGESH-SHOW  LI-LIST
              CA-CATLG/DB2
ITEM:     TB  FG-FILE GRP PS-PSB              D2 (CA ONLY)
                      DL-DLI DBD      TB-SQL TBL      TJ-SQL JOIN
                      VS-VSAM         SQ-SEQ FILE     CQ-CICS QUE
      CJ-CICS JRNL


NAME:    TELON.TRGTASKC___________    (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
DESC:    ___________________________________


BASE:    _______________________      (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                              (USED ONLY FOR FUNCTION: CR)
RDBMS:   _______   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update SQL Table screen. This screen shows all the time-related TLNROW data. You could make changes here, if necessary.

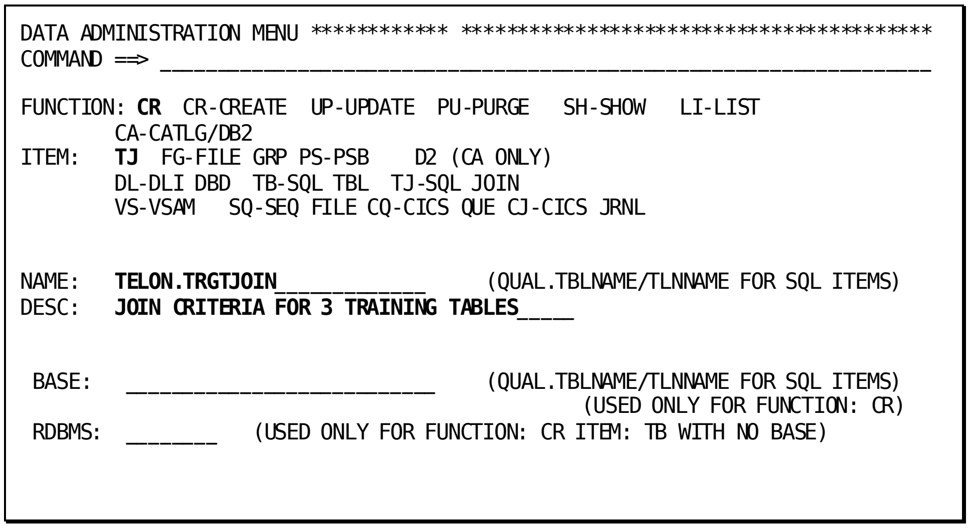


```
UPDATE SQL TABLE ******************** TELON.TRGTIMEC ************ SIZE 000011
COMMAND ==>                            SCROLL ===> CSR
TLNNAME TRGTIMEC DESCR  CREATED BY DB2 CATALOG IMPORT__________  SYNONYM _ Y/N
DCLCOPY TRGTIMEC DCLLBL ___________________________              DCLRDEF _ Y/N
COPY  NONE____      COPYLBL _________________________
       COPYLVI _ Y/N

     COLUMN NAME         ALIAS       KY/AC TYPE LTH DC ^N
****** ****************** ******** TOP OF DATA ********* ** * **** ***** ** *
000001 --TLNROW--         TRGTIMEC                ----
000002 TIME_EMPL_ID       @TIME-EMPL-ID                  CHAR    6       Y
000003 TIME_PROJ_ID       @TIME-PROJ-ID                  CHAR    4       Y
000004 TIME_TASK_ID       @TIME-TASK-ID                  CHAR    4       Y
000005 TIME_YEAR          @TIME-YEAR                     DEC     2       Y
000006 TIME_CHARGES_1_26  @TIME-CHARGES-1-26             CHAR   156      Y
000007 TIME_CHARGES_27_52 @TIME-CHARGES-27-52            CHAR   156      Y
000008 TIME_REG_QRT_TOT1  @TIME-REG-QRT-TOT1       DEC           6  1    Y
000009 TIME_REG_QRT_TOT2  @TIME-REG-QRT-TOT2       DEC           6  1    Y
000010 TIME_REG_QRT_TOT3  @TIME-REG-QRT-TOT3       DEC           6  1    Y
000011 TIME_REG_QRT_TOT4  @TIME-REG-QRT-TOT4       DEC           6  1    Y
****** ****************** ******* BOTTOM OF DATA ******* ** * **** ***** ** *
```

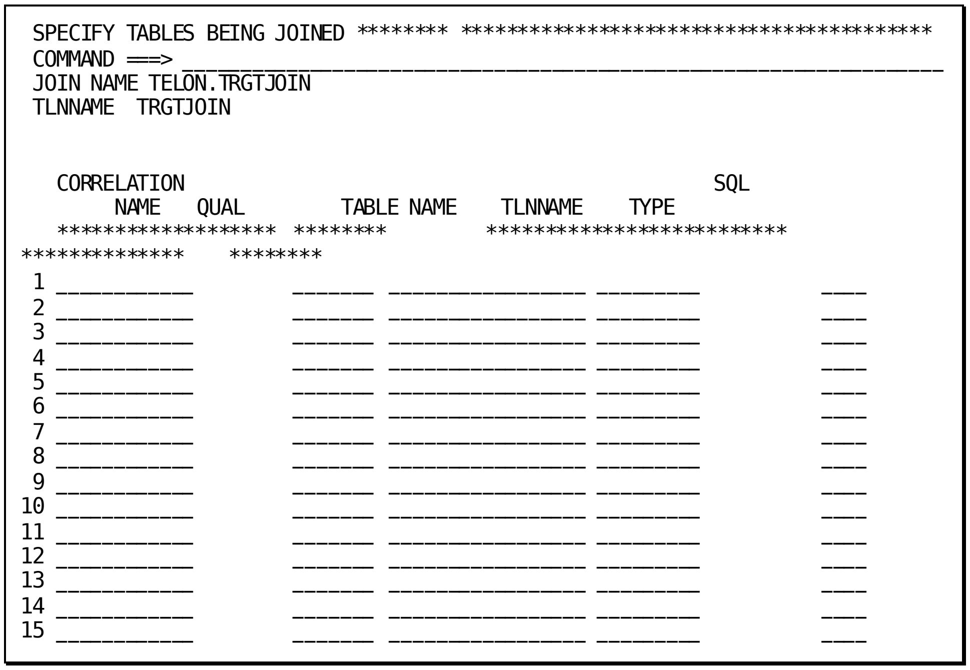
Press PF3 to return to the Data Administration menu.

The next task is to create the Table JOIN criteria.

From the Data Administration menu, type **CR** in the FUNCTION field, **TJ** in the ITEM field, the table name in the NAME field (in this case, **TELON.TRGTJOIN**), a description in the DESC field, and press Enter.


```
DATA ADMINISTRATION MENU ************ ****************************************
COMMAND ==> ___________________________________________________________________

FUNCTION: CR  CR-CREATE  UP-UPDATE  PU-PURGE   SH-SHOW   LI-LIST
        CA-CATLG/DB2
ITEM:   TJ  FG-FILE GRP PS-PSB    D2 (CA ONLY)
        DL-DLI DBD  TB-SQL TBL  TJ-SQL JOIN
        VS-VSAM   SQ-SEQ FILE CQ-CICS QUE CJ-CICS JRNL


NAME:   TELON.TRGTJOIN_____________     (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
DESC:   JOIN CRITERIA FOR 3 TRAINING TABLES____


 BASE:   ___________________________     (QUAL.TBLNAME/TLNNAME FOR SQL ITEMS)
                                                 (USED ONLY FOR FUNCTION: CR)
 RDBMS:  ________   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Specify Tables Being Joined screen. This screen is blank with columns to guide you for input. You can now add new tables to the join.


```
 SPECIFY TABLES BEING JOINED ******** ****************************************
 COMMAND ==> __________________________________________________________________
 JOIN NAME TELON.TRGTJOIN
 TLNNAME  TRGTJOIN


   CORRELATION                                             SQL
        NAME   QUAL       TABLE NAME   TLNNAME    TYPE
   ******************* ********        **************************
**************    ********
 1 ____________        _______ _________________ _________          ____
 2 ____________        _______ _________________ _________          ____
 3 ____________        _______ _________________ _________          ____
 4 ____________        _______ _________________ _________          ____
 5 ____________        _______ _________________ _________          ____
 6 ____________        _______ _________________ _________          ____
 7 ____________        _______ _________________ _________          ____
 8 ____________        _______ _________________ _________          ____
 9 ____________        _______ _________________ _________          ____
10 ____________        _______ _________________ _________          ____
11 ____________        _______ _________________ _________          ____
12 ____________        _______ _________________ _________          ____
13 ____________        _______ _________________ _________          ____
14 ____________        _______ _________________ _________          ____
15 ____________        _______ _________________ _________          ____
```

The next Specify Tables Being Joined screen shows three different tables being input. Notice how you can input different correlation names. This gives you the ability to quickly reference these table names.

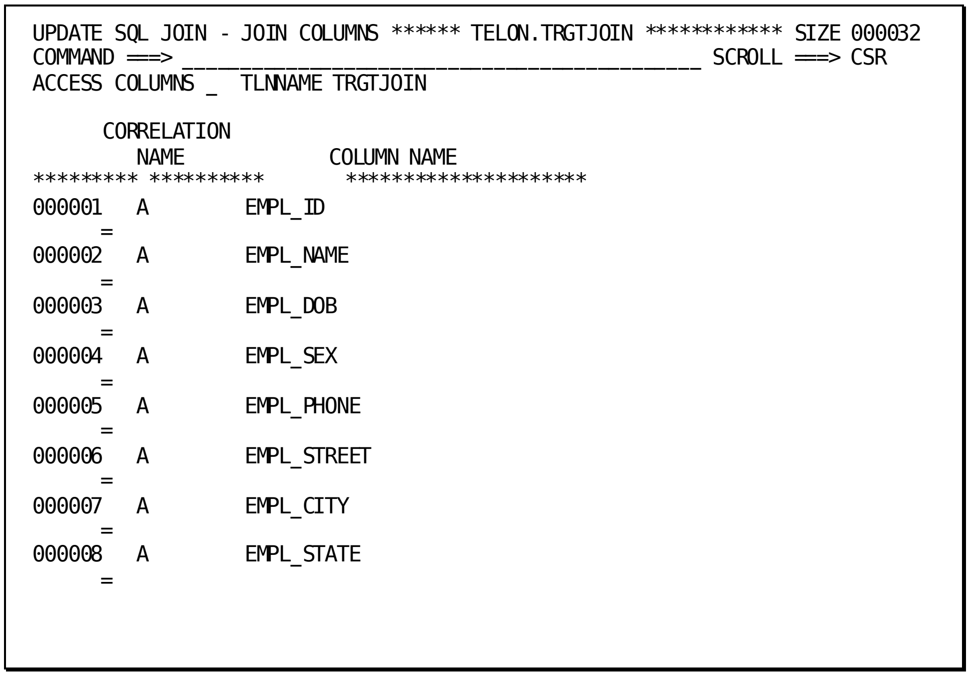
```
 SPECIFY TABLES BEING JOINED ******** ****************************************
 COMMAND ===> _________________________________________________________________
 JOIN NAME TELON.TRGTJOIN
 TLNNAME  TRGTJOIN


  CORRELATION                                              SQL
        NAME   QUAL        TABLE NAME    TLNNAME    TYPE
   ******************* ********         **************************
**************    ********
 1 A___________        TELON_  TRGEMPLC _____ ________            ____
 2 B___________        TELON_  TRGTASKC _____ ________            ____
 3 C___________        TELON_  TRGTIMEC ______ ________           ____
 4 ____________        _______ _________________ _________        ____
 5 ____________        _______ _________________ _________        ____
 6 ____________        _______ _________________ _________        ____
 7 ____________        _______ _________________ _________        ____
 8 ____________        _______ _________________ _________        ____
 9 ____________        _______ _________________ _________        ____
10 ____________        _______ _________________ _________        ____
11 ____________        _______ _________________ _________        ____
12 ____________        _______ _________________ _________        ____
13 ____________        _______ _________________ _________        ____
14 ____________        _______ _________________ _________        ____
15 ____________        _______ _________________ _________        ____
```

After you press Enter, the screen redisplays. CA Telon supplies the TLNNAME associated with the SQL table name. The supplied TLNNAMEs are shown in the Specify Tables Being Joined screen in bold.

```
 SPECIFY TABLES BEING JOINED ******** ****************************************
 COMMAND ===> _________________________________________________________________
 JOIN NAME TELON.TRGTJOIN
 TLNNAME  TRGTJOIN

  CORRELATION                                              SQL
        NAME   QUAL        TABLE NAME    TLNNAME    TYPE
   ******************* ********         **************************
**************    ********
 1 A__________ TELON_  TRGEMPLC ______ TRGEMPLC    DB2_
 2 B__________ TELON_  TRGTASKC ______ TRGTASKC    DB2 _
 3 C__________ TELON_  TRGTIMEC _______ TRGTIMEC    DB2_
 4 ____________        _______ _________________ _________        ____
 5 ____________        _______ _________________ _________        ____
 6 ____________        _______ _________________ _________        ____
 7 ____________        _______ _________________ _________        ____
 8 ____________        _______ _________________ _________        ____
 9 ____________        _______ _________________ _________        ____
10 ____________        _______ _________________ _________        ____
11 ____________        _______ _________________ _________        ____
12 ____________        _______ _________________ _________        ____
13 ____________        _______ _________________ _________        ____
14 ____________        _______ _________________ _________        ____
15 ____________        _______ _________________ _________        ____
```

From the Data Administration menu, you can use the function (UP)DATE with an item of (TJ) TELON join. This allows you to update an existing TELON join.

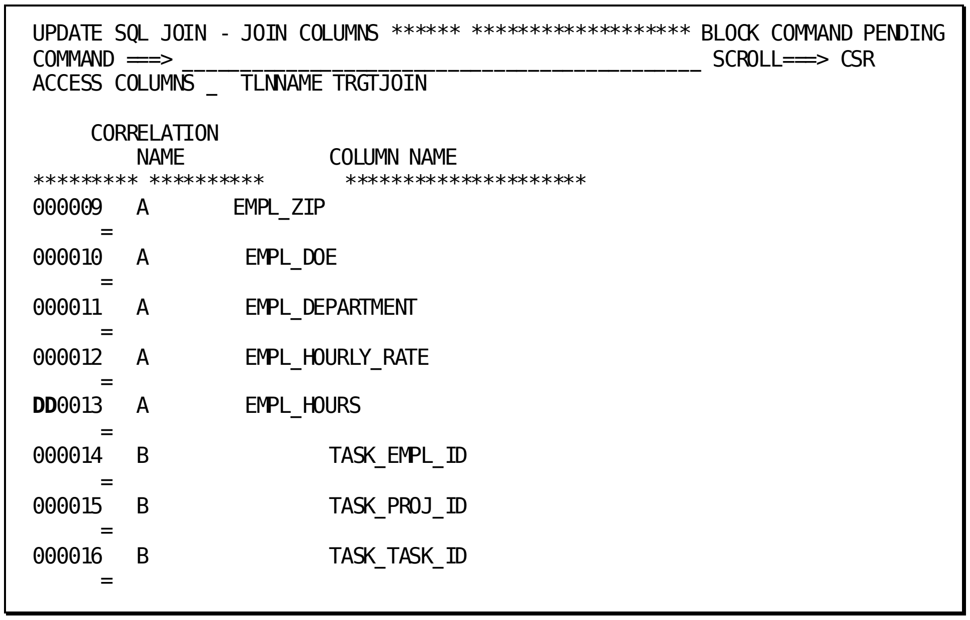
Press PF3 to access the Update SQL Join - Join Columns screen, shown below. Press PF8 to page down. Notice how the correlation names assigned on the Specify Tables Being Joined screen are carried over to the Update SQL Join - Join Columns screen. Use this screen to specify the columns the tables should be joined on.


```
UPDATE SQL JOIN - JOIN COLUMNS ****** TELON.TRGTJOIN ************ SIZE 000032
COMMAND ===> __________________________________________ SCROLL ===> CSR
ACCESS COLUMNS _  TLNNAME TRGTJOIN

      CORRELATION
         NAME            COLUMN NAME
********* **********       ********************
000001   A        EMPL_ID
      =
000002   A        EMPL_NAME
      =
000003   A        EMPL_DOB
      =
000004   A        EMPL_SEX
      =
000005   A        EMPL_PHONE
      =
000006   A        EMPL_STREET
      =
000007   A        EMPL_CITY
      =
000008   A        EMPL_STATE
      =
```

The previous screens show the screen that allows you to update the SQL join columns. Upon entry, the existing join columns are displayed. You can change the existing join columns or add new ones. By entering a non-blank character in the ACCESS COLUMNS field, you can change the columns defined in the TELON join row.

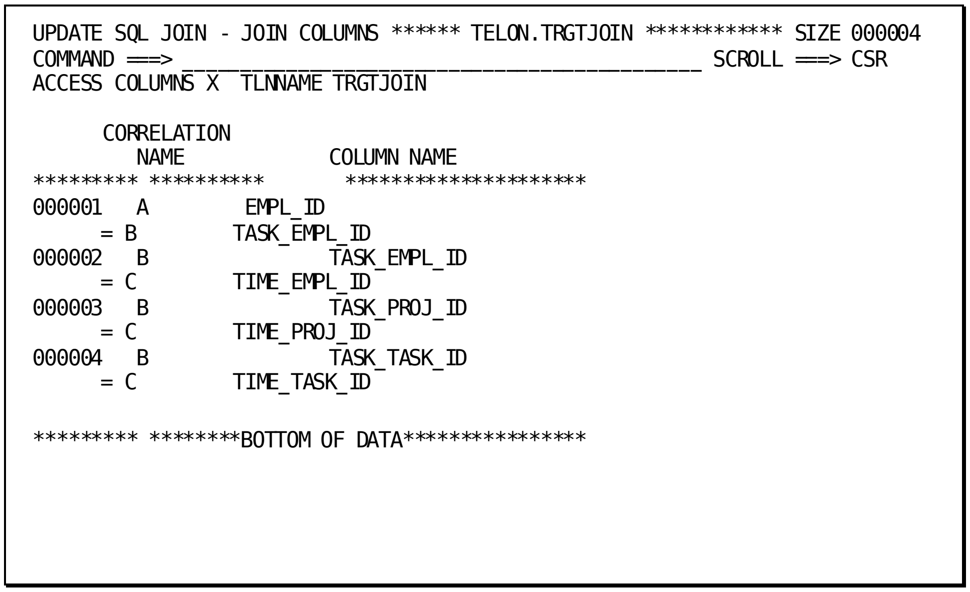
You can edit on the Update SQL Join-Join Columns screen as shown in the next Update SQL Join-Join Columns screen. Type in the correlation names of the tables being joined and the names of the columns being joined on.

You can delete the data that is not used in the correlation.



```
UPDATE SQL JOIN - JOIN COLUMNS ****** ******************* BLOCK COMMAND PENDING
COMMAND ===> ____________________________________________ SCROLL===> CSR
ACCESS COLUMNS _  TLNNAME TRGTJOIN

     CORRELATION
         NAME            COLUMN NAME
********* **********       ********************
000009   A       EMPL_ZIP
      =
000010   A        EMPL_DOE
      =
000011   A        EMPL_DEPARTMENT
      =
000012   A        EMPL_HOURLY_RATE
      =
DD0013   A        EMPL_HOURS
      =
000014   B               TASK_EMPL_ID
      =
000015   B               TASK_PROJ_ID
      =
000016   B               TASK_TASK_ID
      =
```

After editing of all unwanted entries is complete and you press Enter, the screen redisplays with your final correlations of all unwanted entries.



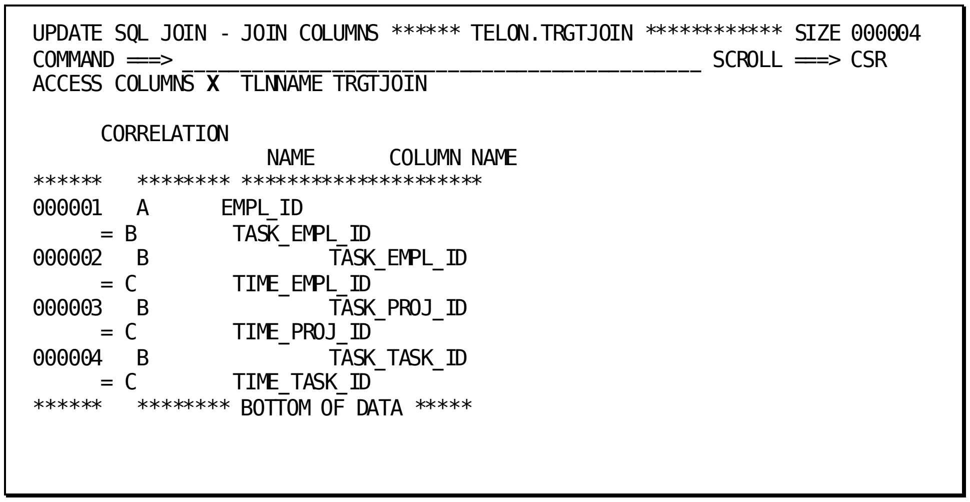
```
UPDATE SQL JOIN - JOIN COLUMNS ****** TELON.TRGTJOIN ************ SIZE 000004
COMMAND ===> ____________________________________________ SCROLL ===> CSR
ACCESS COLUMNS X  TLNNAME TRGTJOIN

      CORRELATION
         NAME            COLUMN NAME
********* **********       ********************
000001   A        EMPL_ID
      = B        TASK_EMPL_ID
000002   B               TASK_EMPL_ID
      = C        TIME_EMPL_ID
000003   B               TASK_PROJ_ID
      = C        TIME_PROJ_ID
000004   B               TASK_TASK_ID
      = C        TIME_TASK_ID

********* ********BOTTOM OF DATA***************
```

This screen sets up a WHERE clause that combines four ways to check data. If the program was generated now, this WHERE clause would look like this:

```
WHERE (A.EMPL_ID = B.TASK_EMPL_ID
AND B.TASK_EMPL_ID = C.TIME_EMPL_ID
AND B.TASK_PROJ_ID = C.TIME_PROJ_ID
AND B.TASK_TASK_ID = C.TIME_TASK_ID)
```

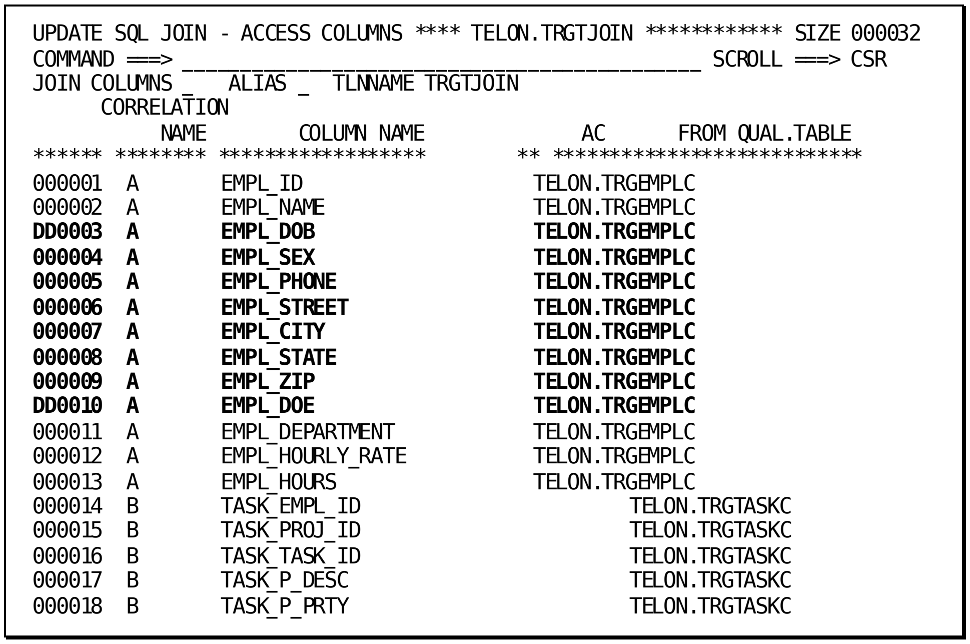
This joining together of tables gives you a way to do data checks from different databases.

Type an **X** (or any non-blank character) in the ACCESS COLUMNS field of the Update SQL Join-Join Columns screen. After you press Enter, CA Telon displays the Update SQL Join-Access Columns screen (note the **X** in the ACCESS COLUMNS field).



```
UPDATE SQL JOIN - JOIN COLUMNS ****** TELON.TRGTJOIN ************ SIZE 000004
COMMAND ===> ___________________________________________ SCROLL ===> CSR
ACCESS COLUMNS X  TLNNAME TRGTJOIN

      CORRELATION
                    NAME      COLUMN NAME
******   ******** *********************
000001   A      EMPL_ID
      = B        TASK_EMPL_ID
000002   B               TASK_EMPL_ID
      = C        TIME_EMPL_ID
000003   B               TASK_PROJ_ID
      = C        TIME_PROJ_ID
000004   B               TASK_TASK_ID
      = C        TIME_TASK_ID
******   ******** BOTTOM OF DATA *****
```

Use the Update SQL Join-Access Columns screen to specify the columns to access in the generated SQL statement for a JOIN. You can edit this screen like the Update SQL Join-Join Columns screen.

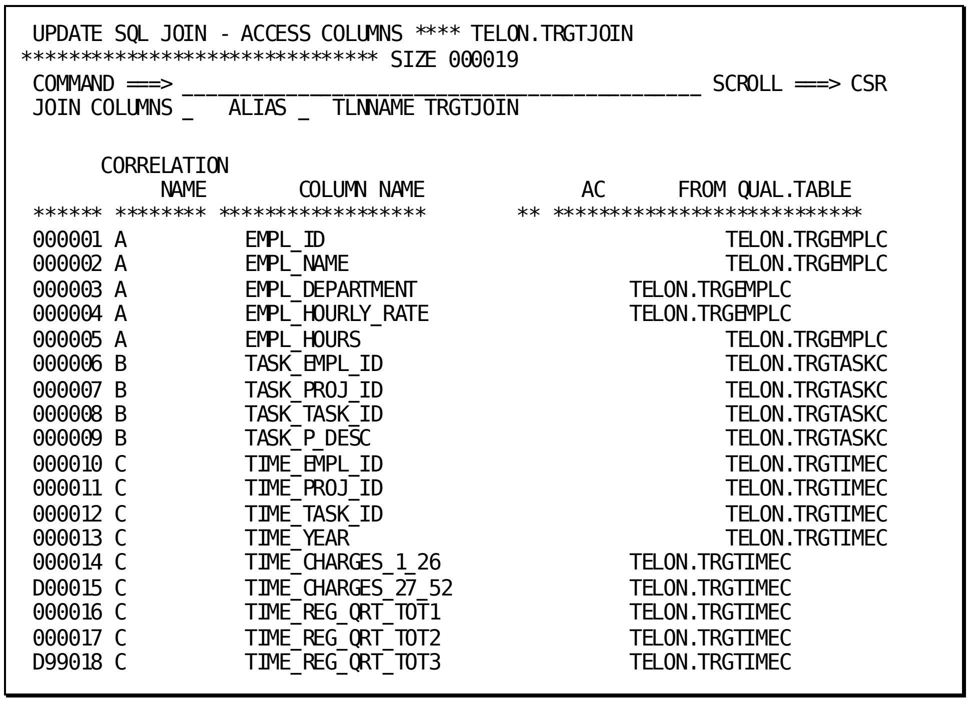


```
UPDATE SQL JOIN - ACCESS COLUMNS **** TELON.TRGTJOIN ************ SIZE 000032
COMMAND ===> _______________________________________________ SCROLL ===> CSR
JOIN COLUMNS _    ALIAS _   TLNNAME TRGTJOIN
      CORRELATION
           NAME        COLUMN NAME              AC      FROM QUAL.TABLE
****** ******** ******************        ** ***************************
000001  A       EMPL_ID                    TELON.TRGEMPLC
000002  A       EMPL_NAME                  TELON.TRGEMPLC
DD0003  A       EMPL_DOB                   TELON.TRGEMPLC
000004  A       EMPL_SEX                   TELON.TRGEMPLC
000005  A       EMPL_PHONE                 TELON.TRGEMPLC
000006  A       EMPL_STREET                TELON.TRGEMPLC
000007  A       EMPL_CITY                  TELON.TRGEMPLC
000008  A       EMPL_STATE                 TELON.TRGEMPLC
000009  A       EMPL_ZIP                   TELON.TRGEMPLC
DD0010  A       EMPL_DOE                   TELON.TRGEMPLC
000011  A       EMPL_DEPARTMENT            TELON.TRGEMPLC
000012  A       EMPL_HOURLY_RATE           TELON.TRGEMPLC
000013  A       EMPL_HOURS                 TELON.TRGEMPLC
000014  B       TASK_EMPL_ID                       TELON.TRGTASKC
000015  B       TASK_PROJ_ID                       TELON.TRGTASKC
000016  B       TASK_TASK_ID                       TELON.TRGTASKC
000017  B       TASK_P_DESC                        TELON.TRGTASKC
000018  B       TASK_P_PRTY                        TELON.TRGTASKC
```

This screen displays a column list of all tables defined in the join. As shown by the highlighted lines, you can delete all unneeded columns.

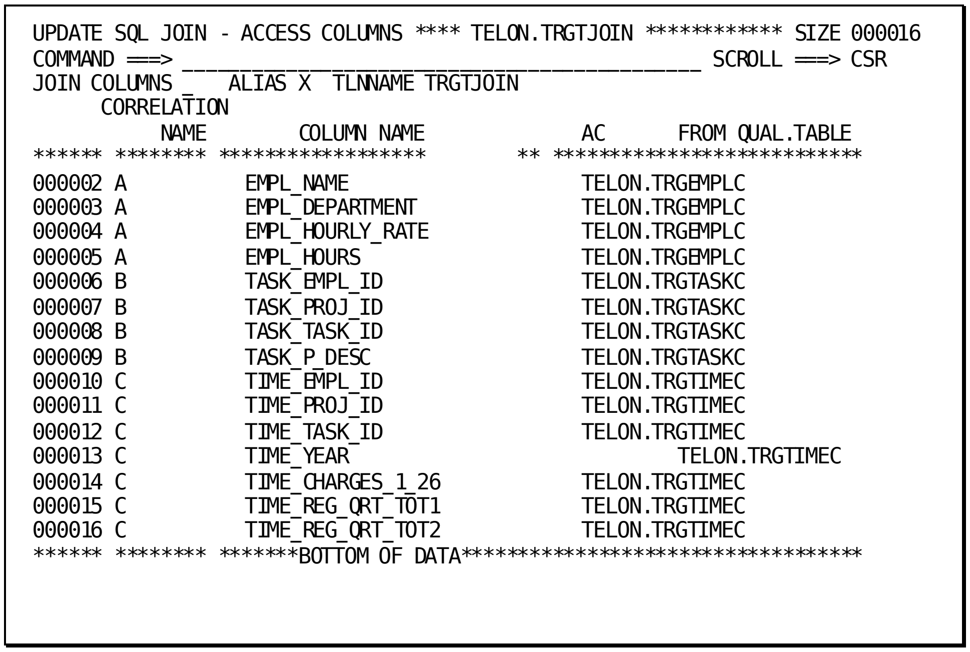
The next Update SQL Join-Access Columns screen shows the continuation of the editing process.

After the editing is complete and you press Enter, you are left with the columns that are accessed in any SQL statement using this criteria. Alternately, all of the columns can be shown as accessible in option 2, but in option 4 in a data group on the Update SQL Detail Data Access screen, you can use the SENCOLs parameter to control which columns are accessed in a generated SQL statement.



```
 UPDATE SQL JOIN - ACCESS COLUMNS **** TELON.TRGTJOIN
******************************* SIZE 000019
 COMMAND ===> _____________________________________________ SCROLL ===> CSR
 JOIN COLUMNS _   ALIAS _  TLNNAME TRGTJOIN


       CORRELATION
            NAME        COLUMN NAME             AC      FROM QUAL.TABLE
 ****** ******** ******************        ** ***************************
 000001 A          EMPL_ID                                  TELON.TRGEMPLC
 000002 A          EMPL_NAME                                TELON.TRGEMPLC
 000003 A          EMPL_DEPARTMENT                 TELON.TRGEMPLC
 000004 A          EMPL_HOURLY_RATE                TELON.TRGEMPLC
 000005 A          EMPL_HOURS                               TELON.TRGEMPLC
 000006 B          TASK_EMPL_ID                             TELON.TRGTASKC
 000007 B          TASK_PROJ_ID                             TELON.TRGTASKC
 000008 B          TASK_TASK_ID                             TELON.TRGTASKC
 000009 B          TASK_P_DESC                              TELON.TRGTASKC
 000010 C          TIME_EMPL_ID                             TELON.TRGTIMEC
 000011 C          TIME_PROJ_ID                             TELON.TRGTIMEC
 000012 C          TIME_TASK_ID                             TELON.TRGTIMEC
 000013 C          TIME_YEAR                                TELON.TRGTIMEC
 000014 C          TIME_CHARGES_1_26               TELON.TRGTIMEC
 D00015 C          TIME_CHARGES_27_52              TELON.TRGTIMEC
 000016 C          TIME_REG_QRT_TOT1               TELON.TRGTIMEC
 000017 C          TIME_REG_QRT_TOT2               TELON.TRGTIMEC
 D99018 C          TIME_REG_QRT_TOT3               TELON.TRGTIMEC
```

Type an **X** (or any non-blank character) in the ALIAS field of the Update SQL Join - Access Columns screen. After you press Enter, CA Telon displays the Add/Update SQL Join Alias screen.



```
UPDATE SQL JOIN - ACCESS COLUMNS **** TELON.TRGTJOIN ************ SIZE 000016
COMMAND ==> ____________________________________________ SCROLL ==> CSR
JOIN COLUMNS _    ALIAS X  TLNNAME TRGTJOIN
      CORRELATION
           NAME        COLUMN NAME             AC      FROM QUAL.TABLE
****** ******** ******************        ** ***************************
000002 A          EMPL_NAME                   TELON.TRGEMPLC
000003 A          EMPL_DEPARTMENT             TELON.TRGEMPLC
000004 A          EMPL_HOURLY_RATE            TELON.TRGEMPLC
000005 A          EMPL_HOURS                  TELON.TRGEMPLC
000006 B          TASK_EMPL_ID                TELON.TRGTASKC
000007 B          TASK_PROJ_ID                TELON.TRGTASKC
000008 B          TASK_TASK_ID                TELON.TRGTASKC
000009 B          TASK_P_DESC                 TELON.TRGTASKC
000010 C          TIME_EMPL_ID                TELON.TRGTIMEC
000011 C          TIME_PROJ_ID                TELON.TRGTIMEC
000012 C          TIME_TASK_ID                TELON.TRGTIMEC
000013 C          TIME_YEAR                           TELON.TRGTIMEC
000014 C          TIME_CHARGES_1_26           TELON.TRGTIMEC
000015 C          TIME_REG_QRT_TOT1           TELON.TRGTIMEC
000016 C          TIME_REG_QRT_TOT2           TELON.TRGTIMEC
****** ******** *******BOTTOM OF DATA***********************************
```

Use the Add/Update SQL Join Alias screen to enter more information about the columns in the generated SQL statement for a JOIN.

In the KEY column, you can enter Y's or numerals. If you enter Y's, the order is from the top of the screen to the bottom. If you specify numerals (1, 2, and so on), you can specify the exact order for column values to be checked in the table. This also controls any "ORDER BY" clause in a generated SQL statement.

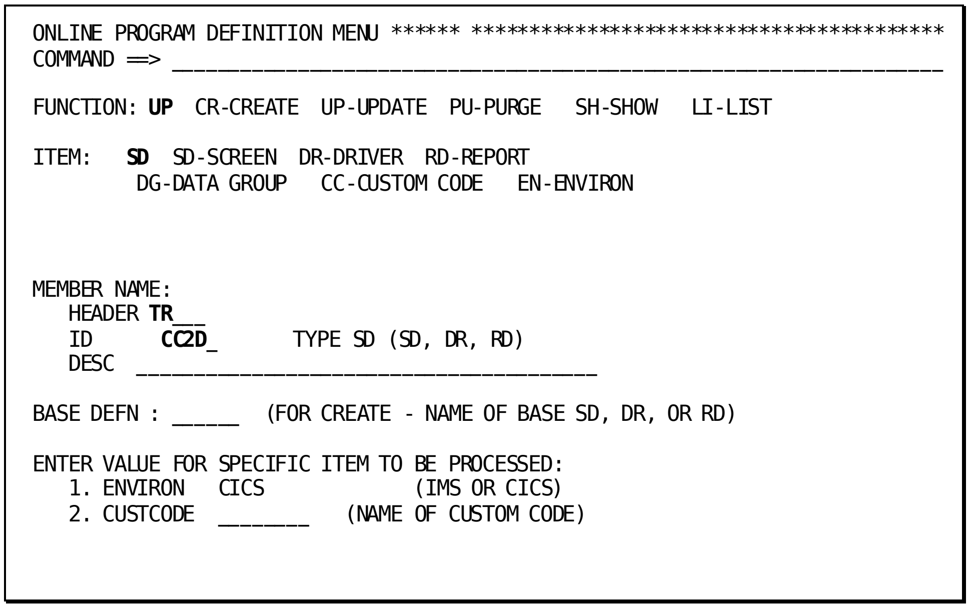
```
 **************** ADD/UPDATE SQL JOIN ALIAS ** ********************************
 COMMAND ==> ____________________________________________ SCROLL ==> 01

               TLNNAME: TRGTJOIN   TABLE: TELON.TRGTJOIN

       CORRELATION
            NAME        COLUMN NAME              AC      FROM QUAL.TABLE
 ****** ******** ******************        ** ***************************
 A                EMPL_ID                        Y_       _
@EMPL-ID_____________________
 A                EMPL_NAME              __      _  @EMPL-NAME_________________
 A                EMPL_DEPARTMENT                __       _
@EMPL-DEPARTMENT____________
 A                EMPL_HOURLY_RATE               __       _
@EMPL-HOURLY-RATE___________
 A                EMPL_HOURS                     __       _
@EMPL-HOURS__________________
 B                TASK_EMPL_ID                   __       _
@TASK-EMPL-ID________________
 B                TASK_PROJ_ID                   Y_       _
@TASK-PROJ-ID________________
 B                TASK_TASK_ID                   Y_       _
@TASK-TASK-ID________________
 B                TASK_P_DESC            __      _
@TASK-P-DESC_________________
 C                TIME_EMPL_ID                   __       _
@TIME-EMPL-ID________________
 C                TIME_PROJ_ID                   __       _
@TIME-PROJ-ID________________
 C                TIME_TASK_ID                   __       _
@TIME-TASK-ID________________
 C                TIME_YEAR                      Y_       _
@TIME-YEAR___________________
 C                TIME_CHARGES_1_26              __       _
@TIME-CHARGES-1-26___________
 C                TIME_REG_QRT_TOT1           __ _  @TIME-REG-QRT-TOT1__________
 C                TIME_REG_QRT_TOT2           __ _  @TIME-REG-QRT-TOT2__________
```

After the KEY information has been typed in and you press Enter, updates to the JOIN criteria are complete.

Type =**4** on the COMMAND line to go to the Online Program Definition menu.

The next example shows CA Telon JOIN criteria being used in a data group.

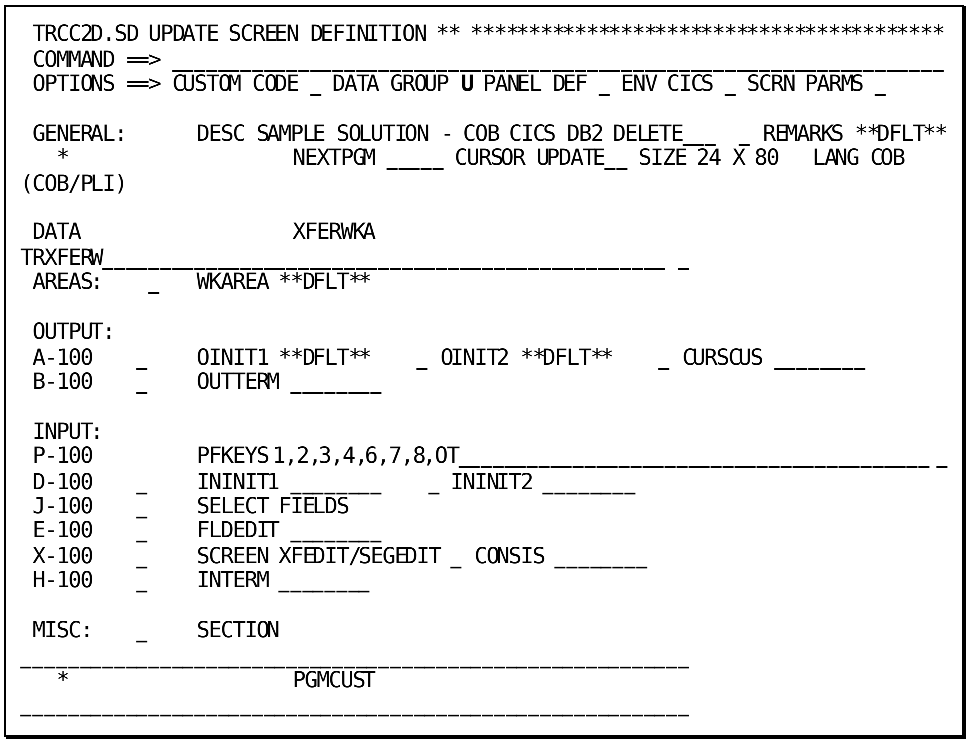
From the Online Program Definition menu, type **UP** in the FUNCTION field, **SD** in the ITEM field, and the appropriate header and ID in the HEADER and ID fields (in this case, **TR** and **CC2D**) and press Enter.



```
ONLINE PROGRAM DEFINITION MENU ****** ****************************************
COMMAND ==> _________________________________________________________________

FUNCTION: UP  CR-CREATE  UP-UPDATE  PU-PURGE   SH-SHOW   LI-LIST

ITEM:   SD  SD-SCREEN  DR-DRIVER  RD-REPORT
         DG-DATA GROUP   CC-CUSTOM CODE   EN-ENVIRON



MEMBER NAME:
   HEADER TR__
   ID      CC2D_      TYPE SD (SD, DR, RD)
   DESC  ________________________________________

BASE DEFN : _____  (FOR CREATE - NAME OF BASE SD, DR, OR RD)

ENTER VALUE FOR SPECIFIC ITEM TO BE PROCESSED:
   1. ENVIRON   CICS             (IMS OR CICS)
   2. CUSTCODE  ________   (NAME OF CUSTOM CODE)
```

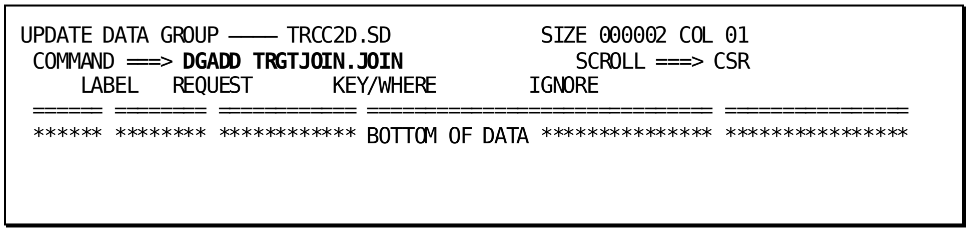
CA Telon displays the Update Screen Definition screen.

Type a non-blank character (in this case, **U**) in the DATA GROUP field to update the data group and press Enter to access the Update Data Group screen.



```
 TRCC2D.SD UPDATE SCREEN DEFINITION ** ****************************************
 COMMAND ==> ___________________________________________________________________
 OPTIONS ==> CUSTOM CODE _ DATA GROUP U PANEL DEF _ ENV CICS _ SCRN PARMS _

 GENERAL:      DESC SAMPLE SOLUTION - COB CICS DB2 DELETE___  _ REMARKS **DFLT**
   *                   NEXTPGM ____ CURSOR UPDATE__ SIZE 24 X 80   LANG COB
(COB/PLI)

 DATA                  XFERWKA
TRXFERW_______________________________________________ _
 AREAS:     _   WKAREA **DFLT**

 OUTPUT:
 A-100     _    OINIT1 **DFLT**    _ OINIT2 **DFLT**    _ CURSCUS _______
 B-100     _    OUTTERM _______

 INPUT:
 P-100          PFKEYS 1,2,3,4,6,7,8,OT_________________________________________ _
 D-100     _    ININIT1 _______    _ ININIT2 _______
 J-100     _    SELECT FIELDS
 E-100     _    FLDEDIT _______
 X-100     _    SCREEN XFEDIT/SEGEDIT _ CONSIS _______
 H-100     _    INTERM _______

 MISC:     _    SECTION
________________________________________________________
   *                   PGMCUST
________________________________________________________
```

From the Update Data Group screen, type the appropriate DGADD command (in this case, **DGADD TRGTJOIN.DJ**) and press Enter.



```
UPDATE DATA GROUP ----- TRCC2D.SD            SIZE 000002 COL 01
 COMMAND ===> DGADD TRGTJOIN.JOIN               SCROLL ===> CSR
     LABEL   REQUEST      KEY/WHERE        IGNORE
 ===== ======= ========== ========================= ===============
 ****** ******** ************ BOTTOM OF DATA *************** ****************
```

CA Telon initializes the Update Data Group screen with the requested information. Type an **I** in the left-hand columns and press Enter to insert a space for a data access request.

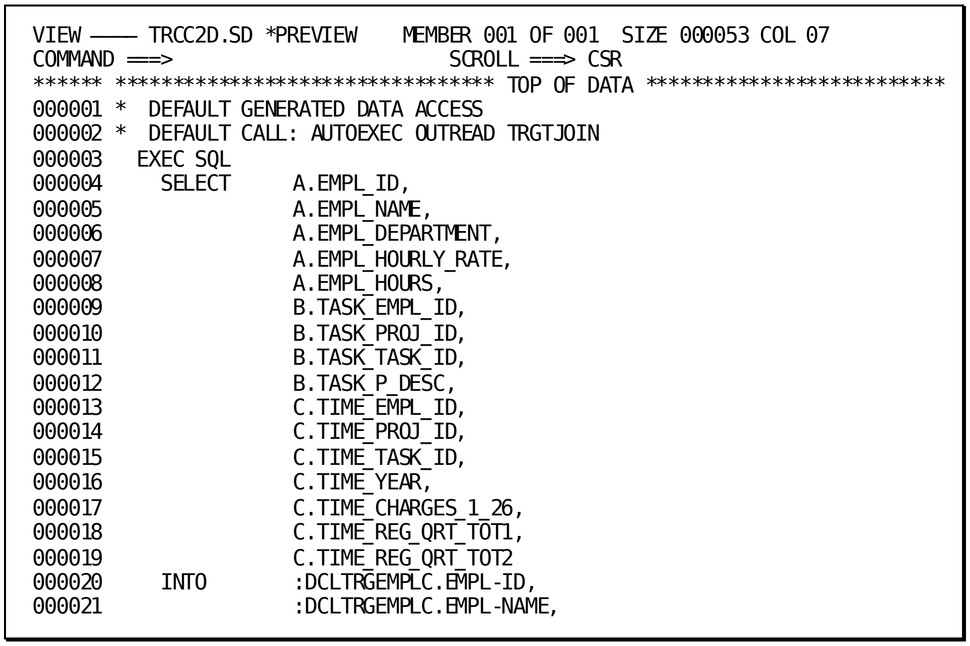
```
UPDATE DATA GROUP ——— TRCC2D.SD SIZE 000003 COL 01
COMMAND ===> SCROLL ===> CSR
LABEL REQUEST KEY/WHERE IGNORE
===== ======= ========== ========================== =============
TAB==> TRGTJOIN * TLNJOIN * TELON.TRGTJOIN
I OW=> TRGTJOIN @DUMMY @EMPL-ID,TASK-PROJ-ID,TASK-TAS
****** ******** ************ BOTTOM OF DATA *************** ****************
```

Type a **V**, **AUTOEXEC**, **OUTREAD**, and **@** in the KEY/WHERE field as shown in the Update Data Group screen.

The @ sign indicates a desire to inherit, as the key columns for this data access request, those columns specified as being key columns in option 2 for this TLNROW. This field can also be left blank or different columns manually entered here could be used as key columns for the generated SQL statement if desired.

```
UPDATE DATA GROUP ——— TRCC2D.SD             SIZE 000003 COL 01
COMMAND ===>                            SCROLL ===> CSR
    LABEL   REQUEST       KEY/WHERE       IGNORE
===== ======= ========== ========================== =============
TAB==> TRGTJOIN * TLNJOIN * TELON.TRGTJOIN
 ROW=> TRGTJOIN @DUMMY    @EMPL-ID,TASK-PROJ-ID,TASK-TAS
V''''' AUTOEXEC OUTREAD   @
****** ******** ************ BOTTOM OF DATA *************** ****************
```

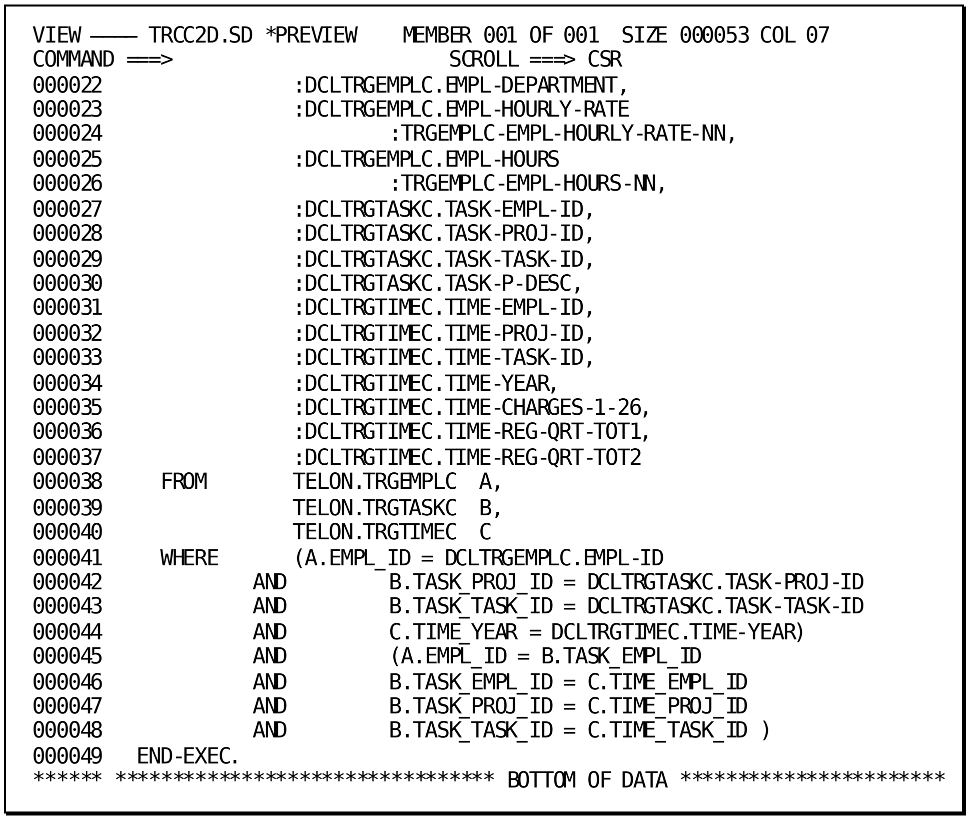
After you press Enter, CA Telon displays the SQL Preview screen. This example shows what the generated code will look like.



```
VIEW ---- TRCC2D.SD *PREVIEW    MEMBER 001 OF 001  SIZE 000053 COL 07
COMMAND ===>                            SCROLL ===> CSR
****** ********************************* TOP OF DATA *************************
000001 *  DEFAULT GENERATED DATA ACCESS
000002 *  DEFAULT CALL: AUTOEXEC OUTREAD TRGTJOIN
000003   EXEC SQL
000004     SELECT     A.EMPL_ID,
000005                A.EMPL_NAME,
000006                A.EMPL_DEPARTMENT,
000007                A.EMPL_HOURLY_RATE,
000008                A.EMPL_HOURS,
000009                B.TASK_EMPL_ID,
000010                B.TASK_PROJ_ID,
000011                B.TASK_TASK_ID,
000012                B.TASK_P_DESC,
000013                C.TIME_EMPL_ID,
000014                C.TIME_PROJ_ID,
000015                C.TIME_TASK_ID,
000016                C.TIME_YEAR,
000017                C.TIME_CHARGES_1_26,
000018                C.TIME_REG_QRT_TOT1,
000019                C.TIME_REG_QRT_TOT2
000020     INTO       :DCLTRGEMPLC.EMPL-ID,
000021                :DCLTRGEMPLC.EMPL-NAME,
```

Press PF8 to page down the screen as shown. The WHERE clause is automatically generated on the basis of the JOIN criteria entered in option 2 for the three tables.

To return to the TDF Main menu, press PF3 three times.



```
VIEW ---- TRCC2D.SD *PREVIEW    MEMBER 001 OF 001  SIZE 000053 COL 07
COMMAND ===>                               SCROLL ===> CSR
000022                    :DCLTRGEMPLC.EMPL-DEPARTMENT,
000023                    :DCLTRGEMPLC.EMPL-HOURLY-RATE
000024                            :TRGEMPLC-EMPL-HOURLY-RATE-NN,
000025                    :DCLTRGEMPLC.EMPL-HOURS
000026                            :TRGEMPLC-EMPL-HOURS-NN,
000027                    :DCLTRGTASKC.TASK-EMPL-ID,
000028                    :DCLTRGTASKC.TASK-PROJ-ID,
000029                    :DCLTRGTASKC.TASK-TASK-ID,
000030                    :DCLTRGTASKC.TASK-P-DESC,
000031                    :DCLTRGTIMEC.TIME-EMPL-ID,
000032                    :DCLTRGTIMEC.TIME-PROJ-ID,
000033                    :DCLTRGTIMEC.TIME-TASK-ID,
000034                    :DCLTRGTIMEC.TIME-YEAR,
000035                    :DCLTRGTIMEC.TIME-CHARGES-1-26,
000036                    :DCLTRGTIMEC.TIME-REG-QRT-TOT1,
000037                    :DCLTRGTIMEC.TIME-REG-QRT-TOT2
000038     FROM           TELON.TRGEMPLC  A,
000039                    TELON.TRGTASKC  B,
000040                    TELON.TRGTIMEC  C
000041     WHERE          (A.EMPL_ID = DCLTRGEMPLC.EMPL-ID
000042             AND            B.TASK_PROJ_ID = DCLTRGTASKC.TASK-PROJ-ID
000043             AND            B.TASK_TASK_ID = DCLTRGTASKC.TASK-TASK-ID
000044             AND            C.TIME_YEAR = DCLTRGTIMEC.TIME-YEAR)
000045             AND            (A.EMPL_ID = B.TASK_EMPL_ID
000046             AND            B.TASK_EMPL_ID = C.TIME_EMPL_ID
000047             AND            B.TASK_PROJ_ID = C.TIME_PROJ_ID
000048             AND            B.TASK_TASK_ID = C.TIME_TASK_ID )
000049   END-EXEC.
****** ********************************* BOTTOM OF DATA **********************
```

Chapter 4: IMS Tasks

This chapter takes you through common CA Telon administrative tasks performed using DL/I.

The first subject, "DL/I Access", describes important concepts pertaining to DL/I access. The later subjects give sample sessions for doing common DL/I tasks, with appropriate explanations.

DL/I Access

The following section describes considerations unique to DL/I processing. For online, the considerations are valid for both CICS and IMS/DC applications. You implement the concepts here into a program using the Create/Update Data Group screen and the detail access method screens entered through the Create/Update Data Group screen.

This topic provides overview conceptual information on DL/I auto exec and user exec data accesses. For more details, refer to the explanation of each TDF screen in the *Design Facility Reference Guide*.

IMS DBD Overview

A CA Telon IMS DBD (database definition) is a definition of an IMS database. CA Telon supports the import, definition, and maintenance of mainframe IMS database descriptions as CA Telon IMS DBDs. CA Telon IMS DBDs are required to build application programs accessing the IMS database.

A CA Telon IMS DBD consists of one or more areas, data sets, segments, and LCHILDs. Areas and data sets provide information and exist only for DBDs imported to CA Telon. An area defines an area within a Fast Path DEDB (Data Entry Database).

A data set defines a data set within a database. A segment defines an IMS database segment and its position in the database hierarchy. An LCHILD defines an IMS database logical child, which defines a relationship between segments in the same or different databases. An LCHILD could be used to define a logical relationship or a secondary index relationship.

GSAM (Generalized Sequential Access Method) DBDs, which can only be imported to CA Telon, consist of only a data set component object and cannot be updated.

IMS PSB Overview

A *CA Telon IMS PSB* (Program Specification Block) describes a view of one or more databases as certain programs would like to see those databases. It identifies the segments that the programs will be allowed to access, and defines the kinds of functions the programs may perform on each segment. If several IMS programs have similar data access requirements, defining a CA Telon IMS PSB can help keep the program's data access synchronized during development and maintenance.

A CA Telon IMS PSB consists of references to one or more IMS Program Communications Blocks (PCB).

An IMS PCB is a structure used for communicating with the IMS system and IMS databases. There are three types of PCBs that may be defined in a file group. They are:

- Teleprocessing PCBs (TP-PCB)
- Database PCBs (DB-PCB)
- GSAM PCBs (GSAM-PCB)
- The first, a *teleprocessing PCB (TP-PCB)*, is used for sending messages through the IMS data communications teleprocessing monitor. The second, a *database PCB (DB-PCB)*, is used for storing and retrieving information from IMS databases. Database PCBs are further composed of one or more *sensitive segments*, that define the segments in the database that this PCB can access. Database PCBs must contain at least one sensitive segment. The third type of PCB, a *Generalized Sequential Access Method PCB (GSAM-PCB),* allows standard sequential files to be manipulated using IMS calls.

CA Telon IMS PSBs are maintained so that their definitions can be used consistently and quickly by other CA Telon functions and/or applications. CA Telon IMS PSBs are usually defined during the internal design or coding phase of a target application.

Data administrators, designers, and programmers all use the CA Telon IMS PSB. Data administrators use the CA Telon IMS PSB because it provides a standardization, tuning, and consistency mechanisms; it increases productivity (reusability); and, it minimizes the learning curve necessary for writing IMS programs. Programmers use the CA Telon IMS PSB to add their own CA Telon IMS PSBs. System administrators can grant (or restrict) the view and update access privileges for this object.

PSB Processing

For CA Telon to generate DL/I calls through the use of auto execs or user execs, the PSBs and DBD's must be created in Data Administration option 2 (if they do not exist) or must be imported (if they already exist).

Access Specification

You specify generated data access for each database to be used by a program on the Create/Update Data Group screen. Each database is identified on a line of the Create/Update Data Group screen and distinguished by the characters PCB==> in the first field on the line. A database line results from:

- The use of a BASE from another program definition
- The use of a DGADD command
- The repeat or copy of another PCB line in this program definition
- The entry of a new PCB in a blank line.

The database line contains two additional fields: the DBD name and the PCB name, if one exists. Note that the titles at the top of the screen refer to the auto exec and user exec lines, not to the database line.

The DBD name identifies the database being accessed. The PCB name is an alternate label for reference to the PCB within the program.

The PCB name was specified in Data Administration option 2 to allow references to the PCB in the program to use a more meaningful name. This alternate label, the PCB name, is important since it can make the names in the generated program more readable and the program more understandable. If the same database is used more than once in your program, you need to use the alternate label to give each usage a unique name.

You identify each segment in a program on a line of the Create/Update Data Group screen. Each segment is distinguished by the characters SEG==> in the first field on the line. A segment line is created at the same time as its associated database.

CA Telon includes each segment in hierarchical order under its database. The segment line identifies the segment for the auto exec and user exec statements which follow it and also establishes the default key values for the parents of a segment which is accessed. The latter is discussed under "Defining a Parent Path" later in this topic.

The segment line contains three additional fields: the segment label, the segment parent key field, and the segment name (if different from the segment label). Note that the titles at the top of the screen refer to the auto exec and user exec lines, not to the segment line.

The label is described for the Create/Update Data Group screen in the *Design Facility* Reference *Guide*. The segment parent key is described under "Defining a Parent Path" later in this topic. The segment name is the name of the segment as specified in the DBD.

You enter each auto exec and user exec statement on a line under the definition for the segment to be accessed. For information on the general parameters for each access definition, see Generated Data Access in the "Data Access Concepts" chapter. These parameters cause the generation of appropriate SSAs and database calls as well as status checking.

If you require more control of a database access, specify lower level parameters to define characteristics for the SSA and/or call statements generated. For example, a READ request normally generates a call with an IMS function code of GU. You can alter the function code to GN by entering GN in the parameter FUNCtion on the Update DL/I Detail Data Access screen.

Secondary Indices

You implement data accesses through secondary indices in CA Telon by defining a PCB (database) for the secondary index path, the same as for normal IMS secondary index access.

You should set up the PCB for the secondary index in the PSB defined in CA Telon Data Administration option 2. Each segment is listed in hierarchical order as defined through the secondary index path. This is the same as is required for setting up SEGMENT statements for a secondary index within a PSB.

You must bring in a PCB list from a PSB defined in CA Telon Administration option 2 or from an existing program definition using the DGADD command. The PCB list must contain the PCB defining the secondary index and each segment that can be accessed through the secondary index path.

You treat any access through that secondary index as if the PCB for that index were a separate database. That is, you define any auto execs or user execs under the appropriate SEG==> line for that PCB.

There are no special CA Telon considerations in setting up the COPY book for segments within the secondary index PCB.

Multiple PCB Positioning

You sometimes need to maintain PCB positioning at different points in the same database at the same time (for example, when both sequential and random access is needed at the same time for the same database). You can accomplish this in CA Telon in the same way as in native applications.

1. You repeat the PCB line for the database, including any SEG lines for segments required during the second set of accesses.
2. Then you define the auto exec and user exec accesses requiring the first positioning under segments for the first PCB, and auto exec and user exec accesses requiring the second positioning under segments for the second PCB.
3. Then you change the request field on the PCB itself. By doing this, the TDF allows the data group to be saved.
4. Since CA Telon usually sets up separate SSAs for segments within each PCB (see "Data Search Criteria" later in this topic for exceptions), you must put a segment label on at least one of the duplicate segment lines. This avoids duplicate names since the names of SSA fields are usually generated from the segment name.

You define a segment label by setting the LABEL parameter on the Update Segment screen. (You access this screen by entering a U on the SEG line.)

When maintaining multiple PCB positions as above, you sometimes need to have two different occurrences of the same segment type in working storage at the same time. For a description of special considerations for this situation, see Multiple I/O Area Handling in the "Data Access Concepts" chapter.

Defining a Parent Path

When defining an IMS segment at any level except the root, you must specify SSA information for all parents in the hierarchical path to the accessed segment. This is normally very simple in CA Telon because, for most functions, the keys identifying parents in the path are the same for all children.

For each SEG line for a database on the Create/Update Data Group screen, a KEY field exists. This KEY field identifies the variable containing the key to be used within SSAs of any parent segments in a data access to any child segments. In other words, you need to consider KEY fields as identifying any parent segments in the hierarchical path for access of a lower level segment.

You must ensure that proper key values are moved to those variables before execution of a lower level auto exec or user exec.

For the above situation, you control the variable name specified in the KEY field in one of two ways:

- You can define the variable name for the default Data Search Criteria (DSC) in Data Administration option 2. If you do this, then the KEY field for the parent segment on the Create/Update Data Group screen is @variable-name, where variable-name is the value entered in Data Administration option 2. If the field is spaces, no variable name was specified in Data Administration option 2. This technique allows you to standardize on KEY field usage in an application.
- You can define the variable name for the default DSC on the Create/Update Data Group screen by keying in the name in the KEY field for each parent segment.

In most cases, you should set up all parent key values in one of the above ways. This means the variable name for parent segments in all data accesses is the same. Though this works for the majority of cases, sometimes you need to have two data accesses use quite different search criteria (SSAs) for the same parent segment. This can occur when you need to override some of the SSA parameters, such as OP CODE, for a parent of the segment being accessed. There are several considerations in handling this situation:

- Set up a data search criteria for the parent segment requiring special handling.
- Specify the use of the data search criterion for the parent on the data access requiring special handling.

  The first consideration, setting up a data search criterion, is described under "Data Search Criteria" later in this topic. For the second consideration, perform the following steps to specify the use of a data search criterion for an auto exec or user exec statement:
- Request the update of detail parameters by entering a **U** in the leftmost column for the appropriate auto exec or user exec line on the Create/Update Data Group screen.
- On the Update DL/I Detail Data Access screen, a line exists identifying the KEY variable name for each parent of the segment being accessed, as well as a line for the segment itself.

You can request that a data search criteria be used to identify special search criteria for any line (parent or the segment itself). You can do this in two ways.

First, if you know the name of the DSC from CA Telon option 2, enter the name in the DLIDSC field on the proper line.

If you do not know the name of the DSC, you can request a list of all existing DSCs for a segment by entering a U in the left most column of the line being defined. The TDF then takes you to the Update Database Data Access SSALIST screen, explained below.

From the Update Database Data Access SSALIST screen, select the appropriate argument by entering an S on the line for that argument. Press Enter to return. If no existing DSC is adequate for your access requirements, you can create a new DSC here.

The above action specifies that a special data search criterion is to be used in defining the SSA for one parent referenced in an auto exec or user exec for a lower level segment. The SSAs for any other parent segments are created as normal.

In addition, the SSAs for the special parent are created as normal for any auto execs or user execs for which the special data search criterion was not referenced.

Setting Path Calls

In CA Telon, you can define path calls in three ways:

- Request a path of segments to be read into a working storage area and then moved into the I/O area for each segment
- Request a path of segments to be read into a group of concatenated I/O areas directly
- Put D command codes within the SSAs for parents of a segment being accessed, accomplishing the same thing as 2, above

Path calls for both the first and second option above are requested by entering the name of the segment at the highest level in the hierarchy to be retrieved in the PATH CALL parameter for the auto exec or user exec of the lowest level segment in the path. This parameter is entered on the Update DL/I Detail Data Access screen. You access this screen from the Create/Update Data Group screen by entering a U beside the auto exec or user exec line.

The only difference between the two options is whether the data is read into a working storage field and then moved into the segment I/O areas, or is read directly into the segment I/O areas. The latter is more efficient, but requires the following:

- The length of the structure in the COPY book must be the exact size of the segment (no additional padding is allowed).
- Only the first COPY book can start as an 01 level.
- All segment I/O areas in the path must be concatenated in the order of the segments in the path.

These three requirements are normal DL/I IMS requirements for path calls.

Whether the data is read into a working storage field and then moved into the segment I/O areas depends on the use of the IOAREA parameter. You specify the IOAREA parameter on the same screen as the PATH parameter explained above.

If you do not override the IOAREA value for the segment which is being accessed with the auto exec or user exec (that is, its value is determined from the IOAREA value specified for the segment in CA Telon Database Administration option 2), then the data is first read into a working storage field and then moved into each segment I/O area.

If you override the IOAREA value so that it is different from that specified in CA Telon Database Administration option 2, then CA Telon assumes that area is the beginning of a group of concatenated segment I/O areas and reads the data directly into that area, without any additional moves. If you blank out the IOAREA parameter, CA Telon assumes that the IOAREA value is not overridden.

For the first option, where the data is first read into a working storage area, the working storage area must be created. This is handled through custom coding.

If the target environment is CICS with EXEC DLI, then CA Telon generates EXEC DLI calls and the working storage area is handled internally within CICS.

If the target environment is not CICS with EXEC DLI, then CA Telon uses the hhUPDTA area as the working storage area. The hhUPDTA area is the working storage area CA Telon uses during most database update processing. For further explanation, refer to the *Programming Concepts Guide*.

If you use the first path call option, you must ensure that the hhUPDTA area is large enough to hold the longest path call, as well as large enough to hold the largest updated segment.

Data Search Criteria

Data search criteria exist in CA Telon to give you more flexibility in controlling the SSAs generated by CA Telon. They also provide a source of information and control for special accesses, such as Boolean reads.

Data search criteria form part of the information carried in CA Telon option 2 for a database segment and are used in option 4 and option 5 to assist in the definition of auto exec and user exec accesses. You do not have to consider them for most calls, but they are invaluable for making complex accesses, such as those with Boolean SSAs.

A data search criterion contains the information necessary to format an SSA, such as DL/I IMS search field name, key length and op codes. It also includes the value of the KEY field for an auto exec or user exec statement. That is, it contains a variable name identifying the source search information to be moved into the SSA.

There are two kinds of data search criteria: default DSCs and regular DSCs. CA Telon often treats the two differently.

- Default DSCs There is one default DSC for every segment within a database. A default DSC is created for each segment when a DBD is imported to CA Telon or when a segment is added to a database in CA Telon Database Administration option 2.

  The default DSC can have a KEY field variable name defined for it. A KEY field variable name exists only if it was entered in Data Administration (option 2). When the KEY field variable name for the default DSC exists, you can override it in the auto exec or user exec statement for the segment being accessed, but not for parents in its path.

- Regular DSC A regular DSC is created only through CA Telon option 2 and always has a KEY field variable name, unless it defines an unqualified access. When the KEY field variable name exists, you can override it in the auto exec or user exec statement, for the segment being accessed, but not for parents in its path.

You use the default DSC for most auto exec and user exec accesses. This DSC is initially assumed when an auto exec or user exec statement is created. The default DSC allows any access to a segment through its normal "key" fields; that is, those fields on which a segment is sequenced.

When accessing through the default DSC, you can request both qualified and unqualified accesses. This is explained in the description for the Update DL/I Detail Data Access screen, provided in the *Design Facility Reference Guide*. CA Telon allows you to override most parameters within the default DSC, but maintenance is easier if you limit modifications to the KEY field and the qualified/unqualified characteristic.

You do not have to specifically request the use of the default DSC. You automatically get its use when you define an auto exec or user exec statement without referencing a regular DSC. You can then override parameters on the auto exec or user exec statement until you make a specific request to use a regular DSC.

Regular DSCs provide more flexibility in the SSAs and calls generated by CA Telon and more control for unusual access paths. Use regular DSCs to specify:

- Boolean logic in accessing a segment.
- A search argument for other than a key field.

- Special op codes or command codes.
- Standardization of variable names used in the KEY field of an auto exec or user exec. When using the default DSC, you can use any variable name to access the associated segment.

Though this has the advantage of being flexible, there is no standardization of variable names used in data access requests. The use of regular DSCs enforces a standard set of variable names (for example, only those defined as regular DSCs).

Your installation sets up any regular DSCs in CA Telon option 2, on the Create/Update SSAs screen. When you are defining an auto exec or user exec which requires a regular DSC, you reference the appropriate DSC that was created in option 2. You do this from the Update DL/I Detail Data Access screen in one of two ways.

- First, if you know the name of the DSC from CA Telon option 2, enter the name in the DLIDSC field on the proper line.
- If you do not know the name of the DSC, you can request a list of all existing DSCs for a segment by entering a U in the leftmost column of the line being defined. In response to that request, you are taken to a list of DSCs that have been defined for the segment being accessed. You select the appropriate DSC by entering an S on the line for that DSC. If no existing DSC is adequate for your access requirements, you can create a new DSC here.

Remember that parameters, including the KEY field, cannot be overridden for a regular DSC.

CICS EXEC DLI Generation (Online Only)

You can request the generation of EXEC DLI calls for DL/I accesses within a CA Telon generated program. The CA Telon source for the generation of EXEC DLI is exactly the same as for normal CALL DL/I. Therefore, the same source generates both EXEC DLI and CALL DL/I programs.

The only cause of the generation of EXEC DLIs is the specification for the CICS target environment. On the screen defining the CICS environment, specify that the DBMS is EXEC DLI. This causes generation of a program with EXEC DLI statements instead of DL/I call statements.

Importing DBDs and PSBs

To import DBDs and PSBs on OS/390 and z/OS, please refer to the *Installation Guide*for information about the procedure.

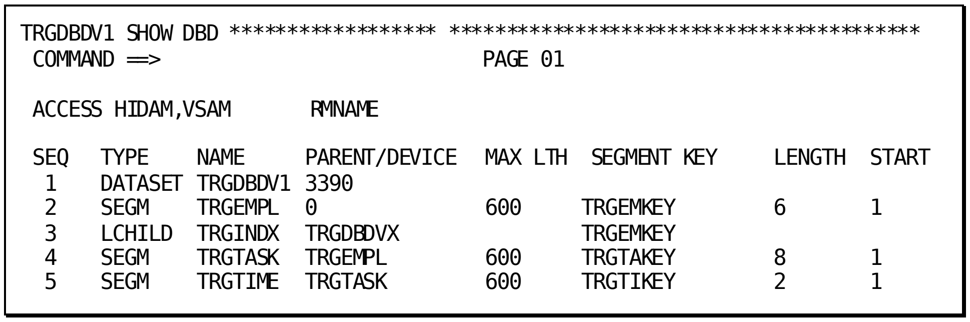
To import DBDs and PSBs on PWS, please refer to the *PWS User Guide* for information about the procedure.

Showing DBDs and PSBs

This topic shows you how to display existing DBDs and PSBs before creating the DBDs and PSBs you need.

Showing DBDs

You can show DBDs from the Data Administration Menu. From this menu, type **SH** in the FUNCTION field, **DL** in the ITEM field, and the DBD Table Name in the NAME field (in this case, **TRGDBDV1**) and press Enter. The Show DBD screen appears for the specified table.



```
TRGDBDV1 SHOW DBD ***************** ***************************************
 COMMAND ==>                                PAGE 01

 ACCESS HIDAM,VSAM       RMNAME

 SEQ   TYPE    NAME      PARENT/DEVICE  MAX LTH  SEGMENT KEY     LENGTH  START
  1    DATASET TRGDBDV1  3390
  2    SEGM    TRGEMPL   0              600     TRGEMKEY        6       1
  3    LCHILD  TRGINDX   TRGDBDVX               TRGEMKEY
  4    SEGM    TRGTASK   TRGEMPL        600     TRGTAKEY        8       1
  5    SEGM    TRGTIME   TRGTASK        600     TRGTIKEY        2       1
```

On the Show DBD screen, you can view IMS database segments. Press PF3 on this screen to return to the Data Administration menu.

Showing PSBs

You can show PSBs from the Data Administration menu. From this menu, type **SH** in the FUNCTION field, **PS** in the ITEM field, and the PSB member name in the NAME field (in this case, **TRPSBCT**) and press Enter. The Show PSB screen appears for the specified table.

```
TRPSBCT  SHOW PSB, FILE GROUP ****** ******************************************
*******************************
 COMMAND ==>
PAGE 01

 LANG  COBOL   COMPAT  YES
                             ***** TP ONLY ******
 SEQ   TYPE    NAME       PCBNAME KEYLEN PROCSEQ  PROCOPT EXP ABC PRT  LTERM
  1    TPPCB              XFER
  2    TPPCB              ABEND                              Y
  3    TPPCB   MPRINT     MPRINT                                       MPRINT
  4    TPPCB              LPRINT
  5    DATABAS TRGDBDV1  EMPLOYEE 16
  6    DATABAS HLPDBDV1  HELP      8
  7    DATABAS HLDDBDV1  HOLD      9
  8    DATABAS WKSDBDV1  WORKSPA   8
```

On the Show PSB, File Group screen you can view file groups (FGs) or, as in this case, IMS Program Specification Blocks (PSBs). Press PF3 on this screen to return to the Data Administration menu.

Creating DBDs and PSBs

There are two ways to create DBDs and PSBs:

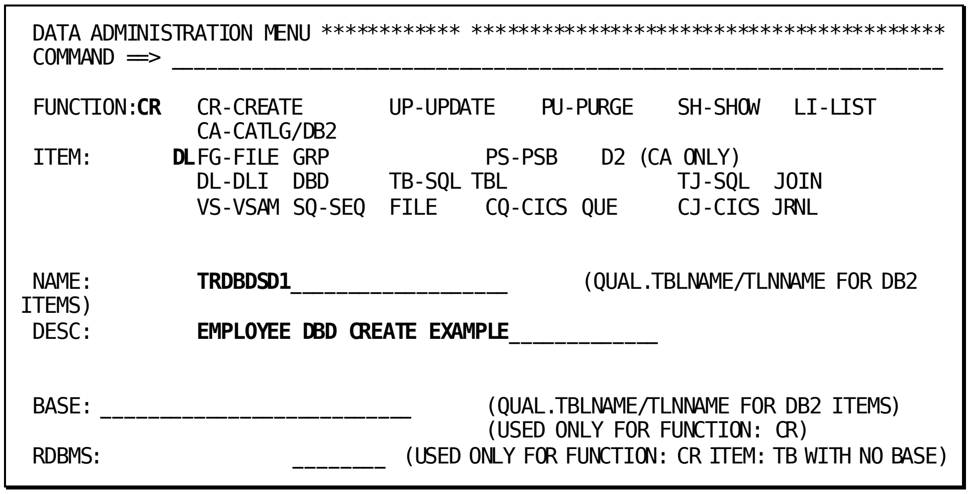
- Typing in a new DBD or PSB
- Using an existing DBD or PSB as a base

Both methods of creating DBDs or PSBs are shown in this topic.

Note: Any created DBD must agree *exactly* with the DL/I DBD or data access will fail.

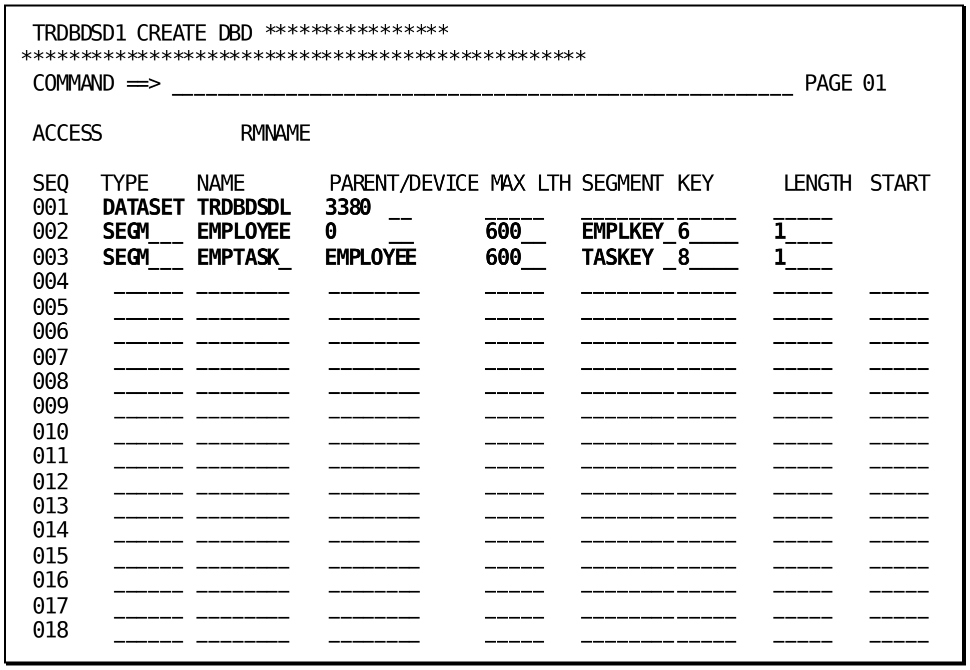
Creating a New DBD

From the Data Administration menu, type **CR** in the FUNCTION field, **DL** in the ITEM field, the DBD name in the NAME field (in this case, **TRDBDSD1**), a description in the DESC field, and press Enter.



```
 DATA ADMINISTRATION MENU ************ ****************************************
 COMMAND ==> __________________________________________________________________

 FUNCTION:CR   CR-CREATE       UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:       DLFG-FILE GRP             PS-PSB    D2 (CA ONLY)
               DL-DLI  DBD     TB-SQL TBL              TJ-SQL  JOIN
               VS-VSAM SQ-SEQ  FILE    CQ-CICS QUE     CJ-CICS JRNL


 NAME:         TRDBDSD1__________________       (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
 DESC:         EMPLOYEE DBD CREATE EXAMPLE____________


 BASE: __________________________      (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)
 RDBMS:                ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays a blank Create DBD screen. Fill in your input for the *DBD as shown in the* Create DBD screen and press Enter. For more information, refer to the Design Facility Reference *Guide* for a full description of the fields on this screen. Press PF3 to save your DBD and return to the Data Administration menu.

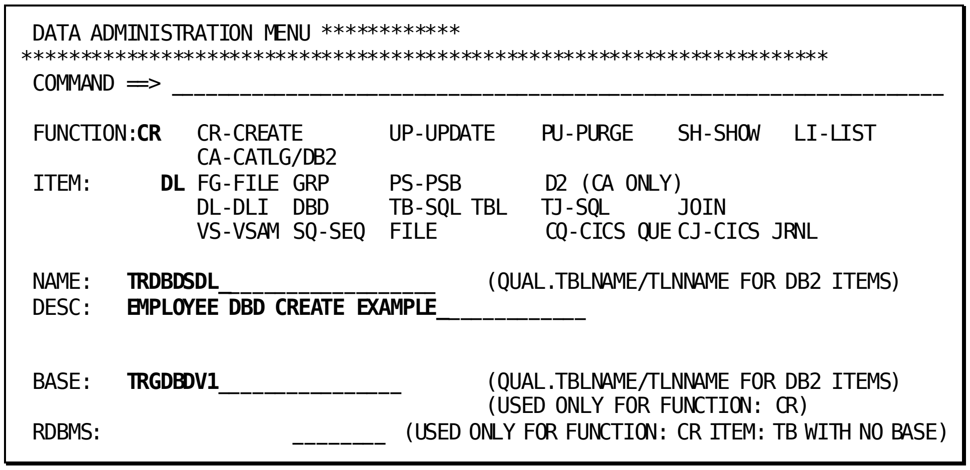


```
 TRDBDSD1 CREATE DBD ****************
************************************************
 COMMAND ==> _____________________________________________________ PAGE 01

 ACCESS           RMNAME

 SEQ   TYPE    NAME       PARENT/DEVICE MAX LTH SEGMENT  KEY      LENGTH  START
 001   DATASET TRDBDSDL   3380  __      _____   ________ _____    _____
 002   SEGM___ EMPLOYEE   0     __      600__   EMPLKEY_6____     1____
 003   SEGM___ EMPTASK_   EMPLOYEE      600__   TASKEY _8____     1____
 004     ______ ________  ________      _____   ________ _____    _____   _____
 005     ______ ________  ________      _____   ________ _____    _____   _____
 006     ______ ________  ________      _____   ________ _____    _____   _____
 007     ______ ________  ________      _____   ________ _____    _____   _____
 008     ______ ________  ________      _____   ________ _____    _____   _____
 009     ______ ________  ________      _____   ________ _____    _____   _____
 010     ______ ________  ________      _____   ________ _____    _____   _____
 011     ______ ________  ________      _____   ________ _____    _____   _____
 012     ______ ________  ________      _____   ________ _____    _____   _____
 013     ______ ________  ________      _____   ________ _____    _____   _____
 014     ______ ________  ________      _____   ________ _____    _____   _____
 015     ______ ________  ________      _____   ________ _____    _____   _____
 016     ______ ________  ________      _____   ________ _____    _____   _____
 017     ______ ________  ________      _____   ________ _____    _____   _____
 018     ______ ________  ________      _____   ________ _____    _____   _____
```

Using an Existing DBD as a Base

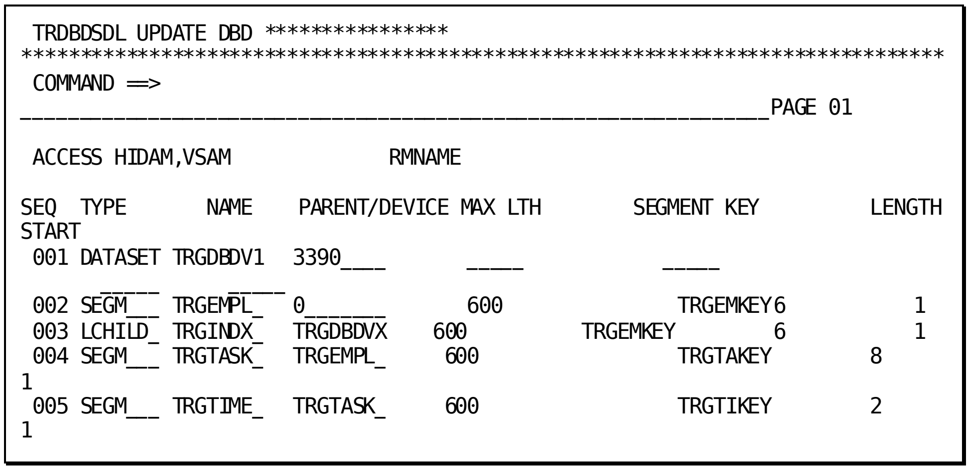
The second way to create a DBD is to use an existing DBD as a base. This procedure saves a tremendous amount of typing if you are creating a large number of similar DBDs.

Type **CR** in the FUNCTION field, **DL** in the ITEM field, the DBD name in the NAME field (in this case, **TRDBDSDL**), a description in the DESC field, an existing DBD name as a base in the BASE field (in this case **TRGDBDV1**) and press Enter.



```
 DATA ADMINISTRATION MENU ************
**********************************************************************
 COMMAND ==> ___________________________________________________________________

 FUNCTION:CR   CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:      DL FG-FILE GRP      PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD      TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ   FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:    TRDBDSDL__________________      (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
 DESC:    EMPLOYEE DBD CREATE EXAMPLE_____________


 BASE:    TRGDBDV1_______________         (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                          (USED ONLY FOR FUNCTION: CR)
 RDBMS:                   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

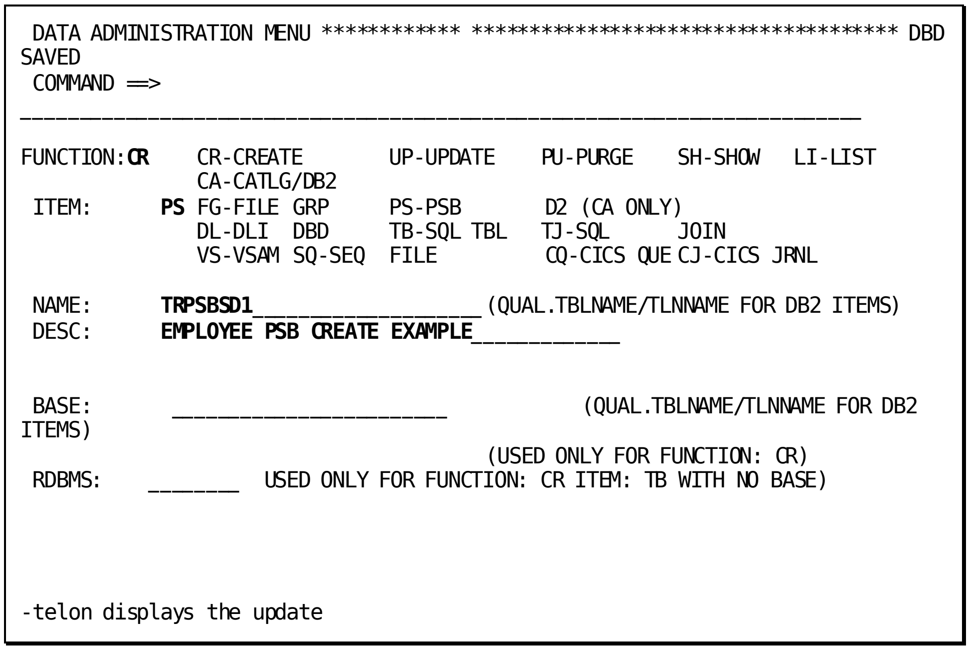
CA Telon displays the Update DBD screen with the DBD information from the base. Press PF3 to save your DBD and return to the Data Administration menu.



```
 TRDBDSDL UPDATE DBD ****************
********************************************************************************
 COMMAND ==>
_________________________________________________________________PAGE 01

 ACCESS HIDAM,VSAM             RMNAME

SEQ  TYPE      NAME    PARENT/DEVICE MAX LTH       SEGMENT KEY         LENGTH
START
 001 DATASET TRGDBDV1   3390____       _____           _____
      _____       _____
 002 SEGM___ TRGEMPL_   0________      600             TRGEMKEY6           1
 003 LCHILD_ TRGINDX_   TRGDBDVX    600         TRGEMKEY        6          1
 004 SEGM___ TRGTASK_   TRGEMPL_     600               TRGTAKEY       8
1
 005 SEGM___ TRGTIME_   TRGTASK_     600               TRGTIKEY       2
1
```

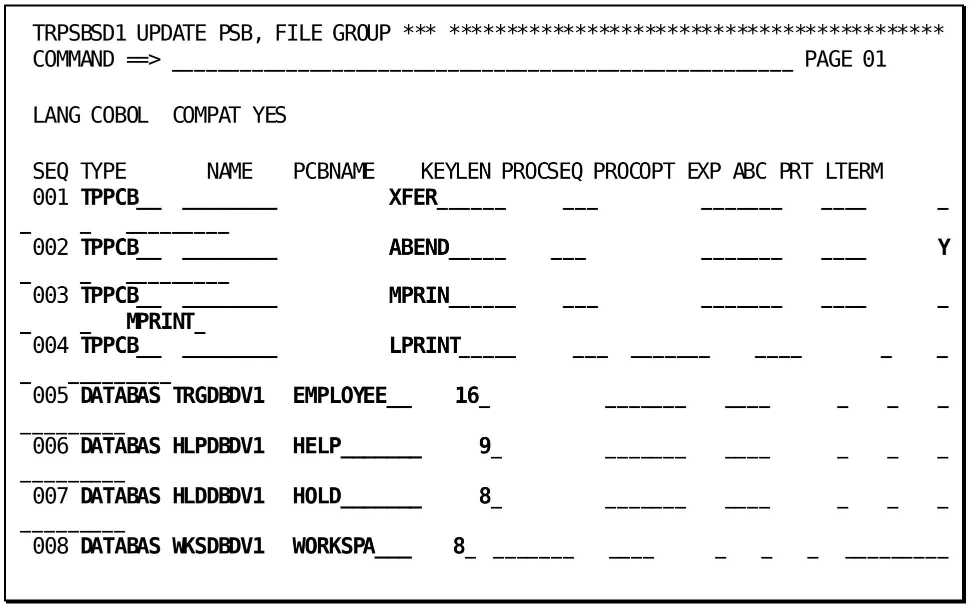
Creating a New PSB

From the Data Administration menu, type **CR** in the FUNCTION field, **PS** in the ITEM field, the PSB name in the NAME field (in this case, **TRPSBSD1**), a description in the DESC field, and press Enter.



```
 DATA ADMINISTRATION MENU ************ ************************************* DBD
SAVED
 COMMAND ==>
______________________________________________________________________

FUNCTION:CR    CR-CREATE       UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:      PS FG-FILE GRP     PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD     TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ  FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:      TRPSBSD1___________________(QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
 DESC:      EMPLOYEE PSB CREATE EXAMPLE____________


 BASE:       _______________________          (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)
 RDBMS:    ________  USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)




-telon displays the update
```

CA Telon displays a blank Create PSB, File Group screen. Fill in your input *for the PSB as* shown in the Update PSB, File Group screen and press Enter. Refer to the Design Facility *Reference Guide* for a full description of the fields on this screen. Press PF3 to save your PSB and return to the Data Administration menu.

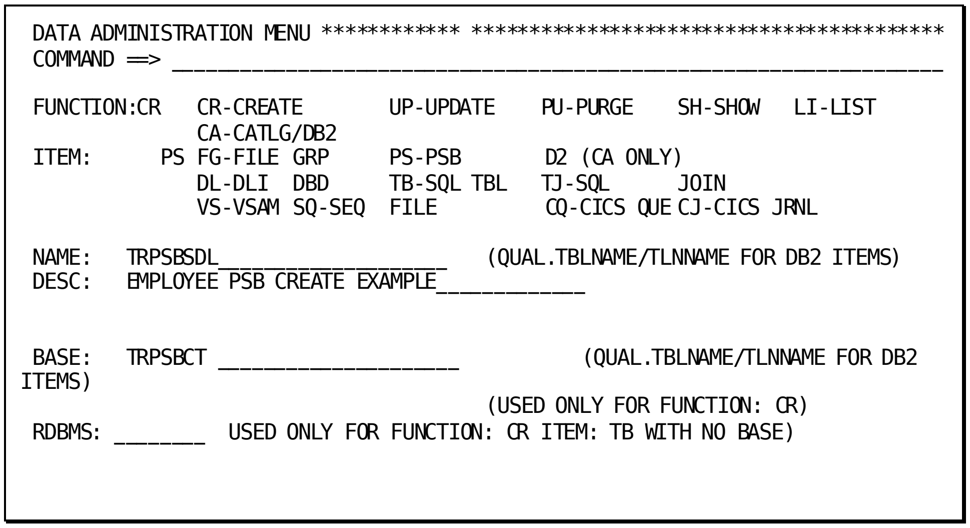


```
 TRPSBSD1 UPDATE PSB, FILE GROUP *** ******************************************
 COMMAND ==> _______________________________________________________ PAGE 01

 LANG COBOL  COMPAT YES

 SEQ TYPE      NAME    PCBNAME    KEYLEN PROCSEQ PROCOPT EXP ABC PRT LTERM
 001 TPPCB__  ________          XFER_____     ___        _______   ____      _
_    _   _________
 002 TPPCB__  ________          ABEND____    ___         _______   ____      Y
_    _   _________
 003 TPPCB__  ________          MPRIN_____    ___        _______   ____      _
_    _   MPRINT_
 004 TPPCB__  ________          LPRINT____     ___  _______    ____       _   _
_   _________
 005 DATABAS TRGDBDV1  EMPLOYEE__    16_          _______   ____     _   _   _
_________
 006 DATABAS HLPDBDV1  HELP_______     9_         _______   ____     _   _   _
_________
 007 DATABAS HLDDBDV1  HOLD_______     8_         _______   ____     _   _   _
_________
 008 DATABAS WKSDBDV1  WORKSPA___    8_ _______   ____     _   _   _  _________
```

Using an Existing PSB as a Base

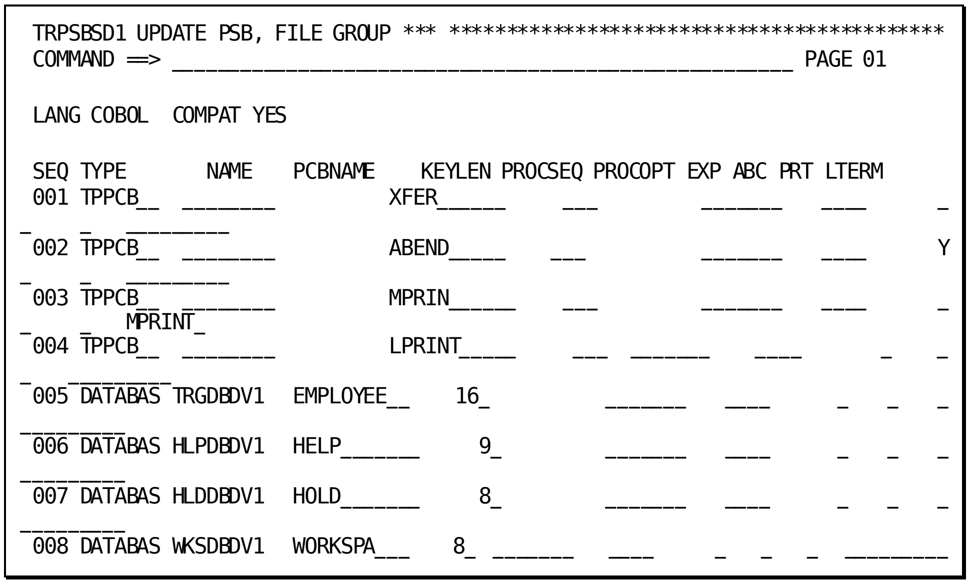
The second way to create a PSB is to use an existing PSB as a base. This procedure saves a tremendous amount of typing if you are creating a large number of similar PSBs.

Type **CR** in the FUNCTION field, **PS** in the ITEM field, the PSB name in the NAME field (in this case, **TRPSBSDL**), a description in the DESC field, an existing PSB name as a base in the BASE field (in this case, **TRPSBCT**) and press Enter.



```
 DATA ADMINISTRATION MENU ************ *****************************************
 COMMAND ==> ____________________________________________________________________

 FUNCTION:CR   CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:      PS FG-FILE GRP      PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD      TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ   FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:    TRPSBSDL____________________    (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
 DESC:    EMPLOYEE PSB CREATE EXAMPLE_____________


 BASE:    TRPSBCT _____________________           (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                          (USED ONLY FOR FUNCTION: CR)
 RDBMS: ________  USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

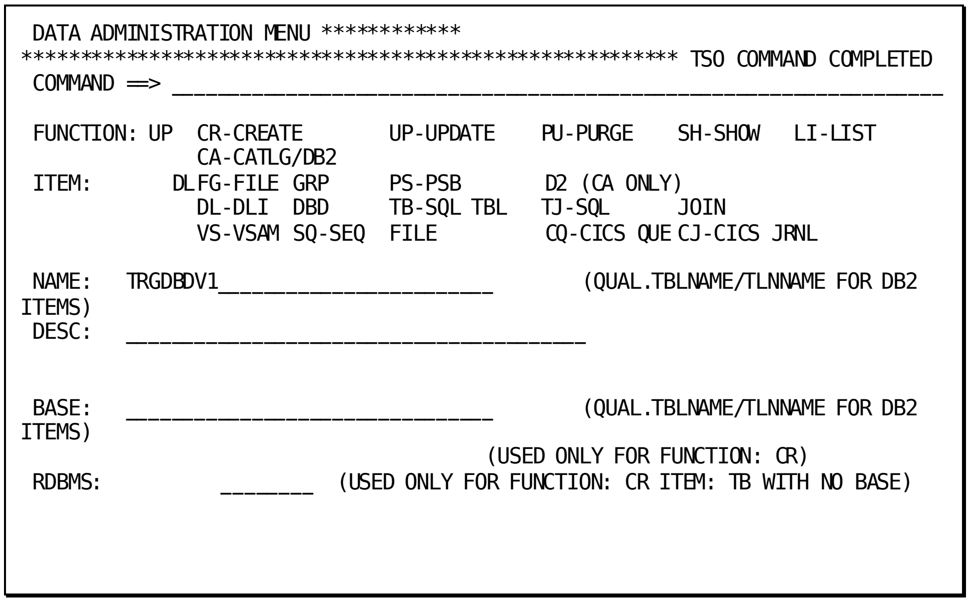
CA Telon displays the Update PSB, File Group screen with the PSB information from the base. Press PF3 to save your PSB and return to the Data Administration menu.



```
 TRPSBSD1 UPDATE PSB, FILE GROUP *** *******************************************
 COMMAND ==> _____________________________________________________ PAGE 01

 LANG COBOL  COMPAT YES

 SEQ TYPE       NAME    PCBNAME    KEYLEN PROCSEQ PROCOPT EXP ABC PRT LTERM
 001 TPPCB__  ________          XFER______    ___         _______   ____      _
_   _   _________
 002 TPPCB__  ________          ABEND_____   ___          _______   ____      Y
_   _   _________
 003 TPPCB__  ________          MPRIN______   ___         _______   ____      _
_   _   MPRINT_
 004 TPPCB__  ________          LPRINT_____    ___  _______    ____       _   _
_ _________
 005 DATABAS TRGDBDV1  EMPLOYEE__    16_        _______   ____      _   _   _
_________
 006 DATABAS HLPDBDV1  HELP_______     9_       _______   ____      _   _   _
_________
 007 DATABAS HLDDBDV1  HOLD_______     8_       _______   ____      _   _   _
_________
 008 DATABAS WKSDBDV1  WORKSPA___    8_ _______   ____     _   _   _  _________
```

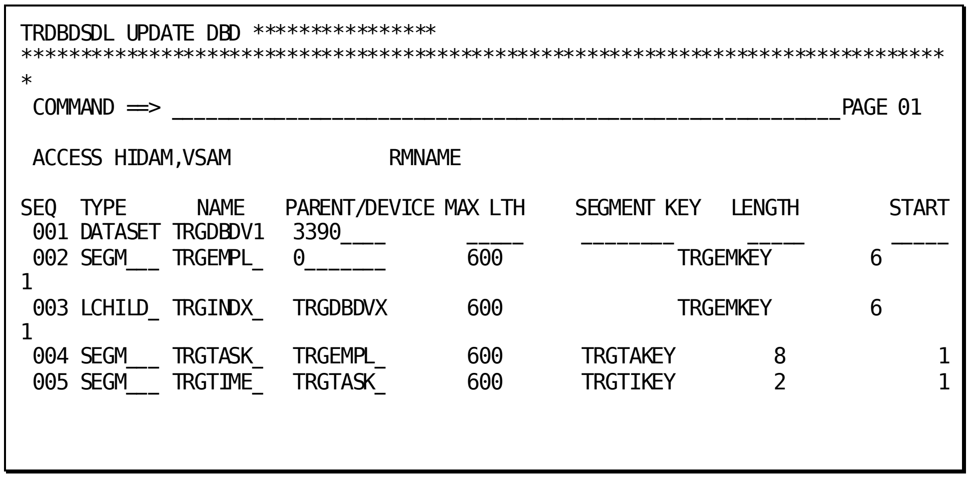
Updating DBDs

From the Data Administration menu, type **UP** in the FUNCTION field, **DL** in the ITEM field, the DBD name in the NAME field (in this case, **TRGDBDV1**) and press Enter.



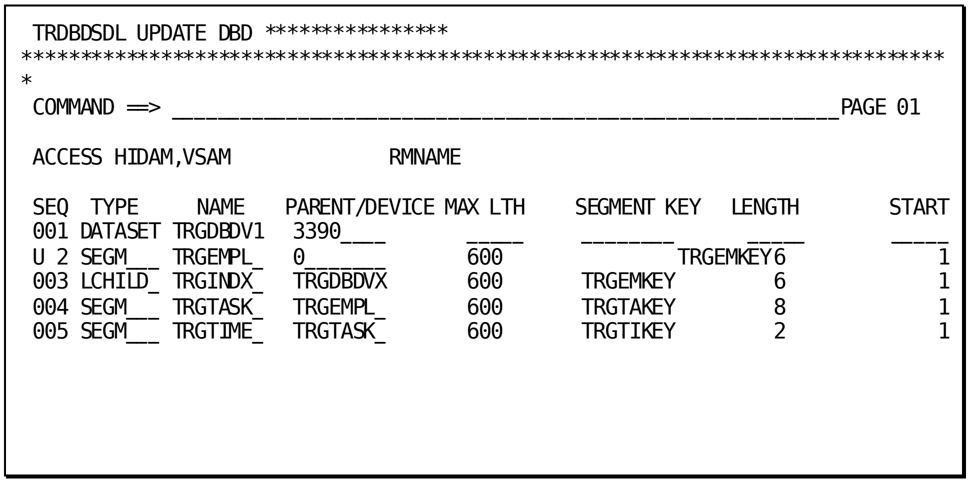
```
 DATA ADMINISTRATION MENU ************
******************************************************** TSO COMMAND COMPLETED
 COMMAND ==> __________________________________________________________________

 FUNCTION: UP  CR-CREATE       UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:       DLFG-FILE GRP     PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD     TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ  FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:    TRGDBDV1________________________       (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
 DESC:    ________________________________________


 BASE:    ________________________________       (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                         (USED ONLY FOR FUNCTION: CR)
 RDBMS:           ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update DBD screen with the DBD information of the requested DBD name (for TRGDBDV1).

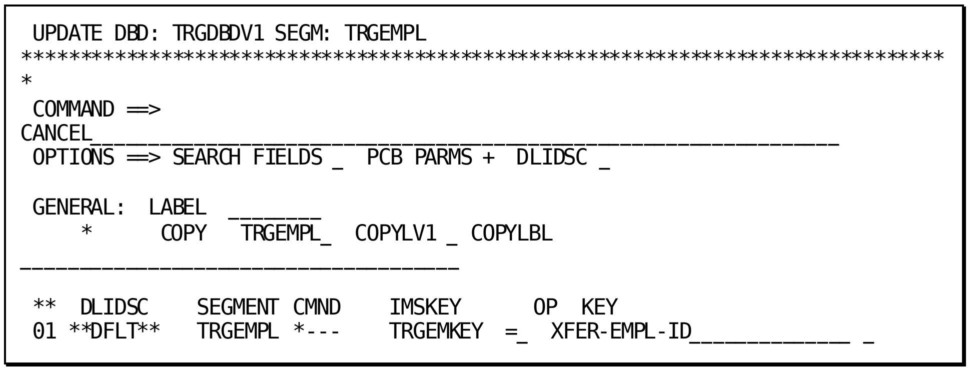


```
TRDBDSDL UPDATE DBD ****************
********************************************************************************
*
 COMMAND ==> __________________________________________________________PAGE 01

 ACCESS HIDAM,VSAM             RMNAME

SEQ  TYPE      NAME    PARENT/DEVICE MAX LTH    SEGMENT KEY  LENGTH       START
 001 DATASET TRGDBDV1   3390____      _____     ________     _____        _____
 002 SEGM___ TRGEMPL_   0_______      600               TRGEMKEY        6
1
 003 LCHILD_ TRGINDX_   TRGDBDVX      600               TRGEMKEY        6
1
 004 SEGM___ TRGTASK_   TRGEMPL_      600      TRGTAKEY        8              1
 005 SEGM___ TRGTIME_   TRGTASK_      600      TRGTIKEY        2              1
```

From the Update DBD screen you can do updates of the segments. For example, the Update DBD screen shows the update of the TRGEMPL segment. Type **U** in the left-hand column of the TRGEMPL segment and press Enter.

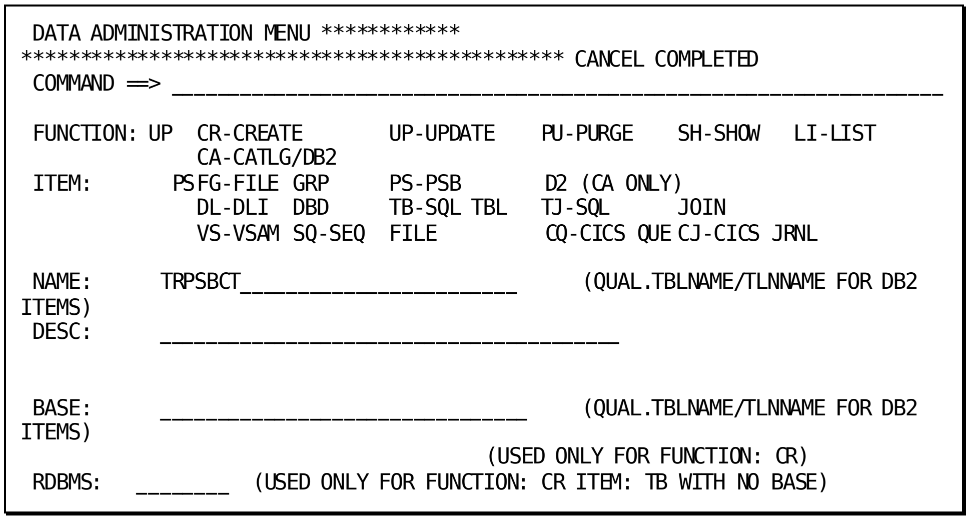

```
 TRDBDSDL UPDATE DBD ****************
********************************************************************************
*
 COMMAND ==> _________________________________________________________PAGE 01

 ACCESS HIDAM,VSAM             RMNAME

 SEQ  TYPE     NAME    PARENT/DEVICE MAX LTH    SEGMENT KEY   LENGTH       START
 001 DATASET TRGDBDV1  3390____       _____     ________     _____       _____
 U 2 SEGM___ TRGEMPL_  0_______       600               TRGEMKEY6             1
 003 LCHILD_ TRGINDX_  TRGDBDVX       600       TRGEMKEY        6             1
 004 SEGM___ TRGTASK_  TRGEMPL_       600       TRGTAKEY        8             1
 005 SEGM___ TRGTIME_  TRGTASK_       600       TRGTIKEY        2             1
```

CA Telon redisplays the Update DBD screen with information for the requested segment. Type **CANCEL** in the COMMAND line to return to the Data Administration menu.


```
 UPDATE DBD: TRGDBDV1 SEGM: TRGEMPL
********************************************************************************
*
 COMMAND ==>
CANCEL________________________________________________________________
 OPTIONS ==> SEARCH FIELDS _  PCB PARMS +  DLIDSC _

 GENERAL:  LABEL  ________
     *      COPY   TRGEMPL_  COPYLV1 _ COPYLBL
______________________________________

 **  DLIDSC    SEGMENT CMND    IMSKEY      OP  KEY
 01 **DFLT**   TRGEMPL *---    TRGEMKEY  =_  XFER-EMPL-ID_____________ _
```

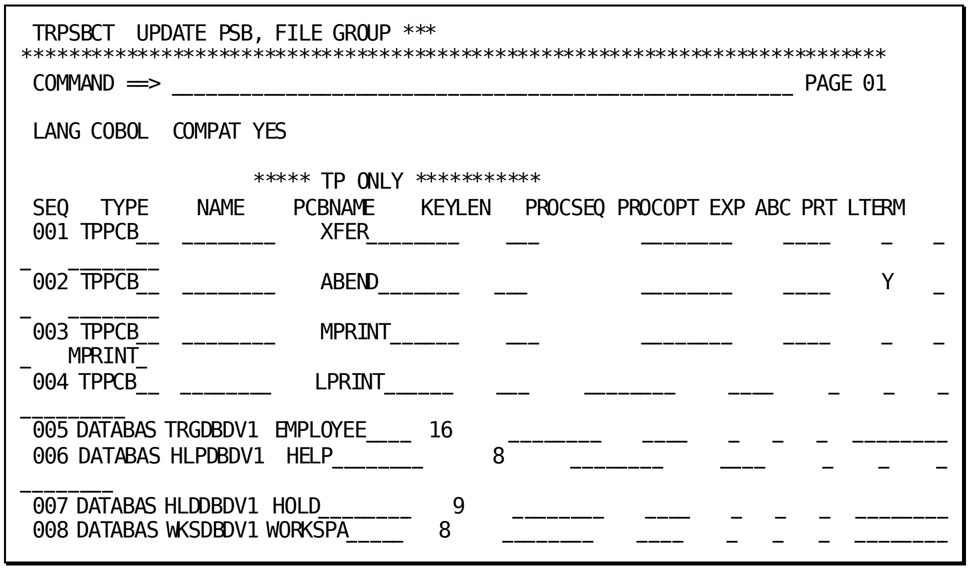
Updating PSBs

From the Data Administration menu, type **UP** in the FUNCTION field, **PS** in the ITEM field, the PSB name in the NAME field (in this case, **TRPSBCT**) and press Enter.



```
 DATA ADMINISTRATION MENU ************
*********************************************** CANCEL COMPLETED
 COMMAND ==> ___________________________________________________________________

 FUNCTION: UP  CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:       PSFG-FILE  GRP     PS-PSB       D2 (CA ONLY)
               DL-DLI   DBD     TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM  SQ-SEQ  FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:      TRPSBCT________________________     (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
 DESC:      ________________________________________


 BASE:      ________________________________     (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                         (USED ONLY FOR FUNCTION: CR)
 RDBMS:   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update PSB, File Group screen with the PSB information of the requested PSB name (for TRPSBCT).



```
 TRPSBCT  UPDATE PSB, FILE GROUP ***
***************************************************************************
 COMMAND ==> ______________________________________________________ PAGE 01

 LANG COBOL  COMPAT YES

                   ***** TP ONLY ***********
 SEQ   TYPE     NAME     PCBNAME    KEYLEN   PROCSEQ PROCOPT EXP ABC PRT LTERM
 001 TPPCB__  ________    XFER________   ___         ________    ____    _    _
_  ________
 002 TPPCB__  ________    ABEND_______  ___          ________    ____    Y    _
_  ________
 003 TPPCB__  ________    MPRINT______   ___         ________    ____    _    _
_   MPRINT_
 004 TPPCB__  ________   LPRINT______   ___    ________    ____    _    _    _
_________
 005 DATABAS TRGDBDV1 EMPLOYEE____  16     ________   ____    _   _   _  ________
 006 DATABAS HLPDBDV1  HELP________      8     ________    ____    _    _    _
________
 007 DATABAS HLDDBDV1 HOLD________    9    ________   ____    _   _   _  ________
 008 DATABAS WKSDBDV1 WORKSPA_____   8    ________   ____    _   _   _  ________
```

From the Update PSB, File Group screen you can do updates of the IMS Program Specification Blocks. For example, the Update PSB, File Group screen shows the update of the Employee PCB name. Type **U** in the left-hand column of the EMPLOYEE PCB name and press Enter.

Note: Changes must mirror changes in DL/I PCBs or data access will fail.

```
 TRPSBCT  UPDATE PSB, FILE GROUP ***
***************************************************************************
 COMMAND ==> ______________________________________________________ PAGE 01

 LANG COBOL  COMPAT YES
                             ***** TP ONLY ******
 SEQ TYPE   NAME   PCBNAME  KEYLEN PROCSEQ PROCOPT EXP ABC PRT LTERM
 001 TPPCB__ ________ XFER________ ___ ________  ____  _  _  _ ________
 002 TPPCB__ ________ ABEND_______ ___ ________  ____  Y  _  _ ________
 003 TPPCB__ ________ MPRINT______ ___ ________  ____  _  _  _ MPRINT__
 004 TPPCB__ ________ LPRINT______ ___ ________  ____  _  _  _ ________
 U 5 DATABAS TRGDBDV1 EMPLOYEE____  16 ________  ____  _  _  _ ________
 006 DATABAS HLPDBDV1 HELP________  8 ________  ____  _  _  _ ________
 007 DATABAS HLDDBDV1 HOLD________  9 ________  ____  _  _  _ ________
 008 DATABAS WKSDBDV1 WORKSPA_____  8 ________  ____  _  _  _ ________
```

CA Telon displays the Update Sensitive Segment screen with information for the requested PCB name. You can then update information on this screen.

Note: You can use the COMMAND line to enter the INIT command. This command causes CA Telon to reinitialize segment characteristics for the database. Even if only one segment is being displayed, INIT reinitializes all of the segments on the database.

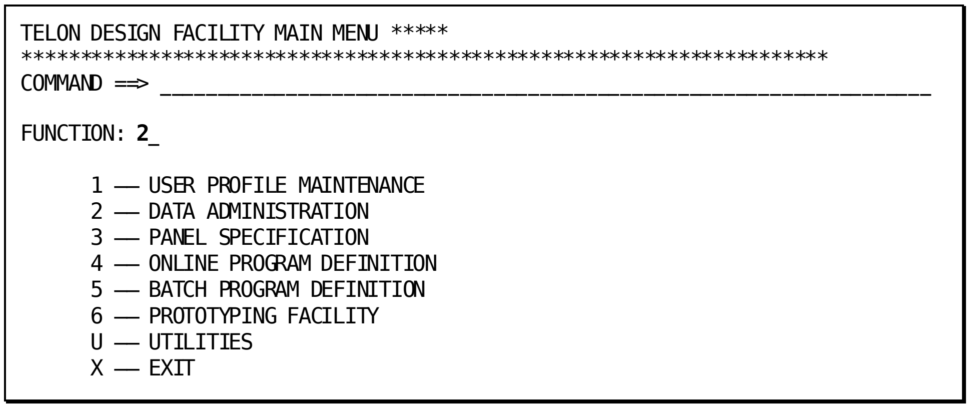
Press PF3 to save this information and to return to the update PSB, File Group screen. Press PF3 to return to the Data Administration menu. Press PF3 to return to the TDF Main menu.

```
 TRPSBCT  UPDATE SENSITIVE SEG ******
***************************************************************************
 COMMAND ==> ______________________________________________________ PAGE 01

 PCBNAME EMPLOYEE      DBNAME TRGDBDV1

 SEQ  NAME       LABEL  PROCOPT           INDICES
 001 TRGEMPL_  ________        A____
      ___________________________________________________
 002 TRGTASK_  ________        A____
      ___________________________________________________
 003 TRGTIME_  ________        A____
      ___________________________________________________
```

Updating DL/I DSCs (Boolean Example)

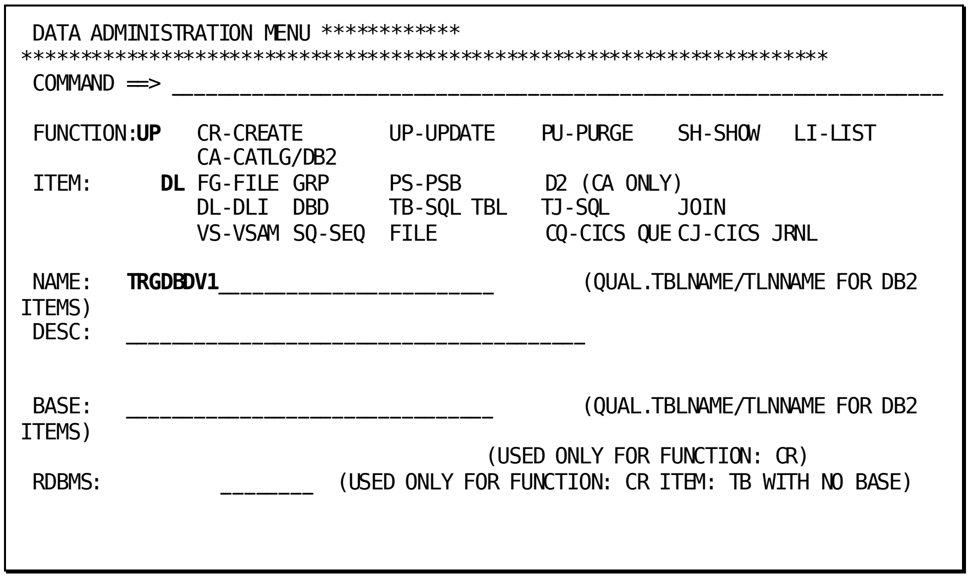
This topic shows you how to update DL/I DSCs using a Boolean example.

From the TDF Main menu, type **2** in the FUNCTION field and press Enter. CA Telon displays the Data Administration menu.



```
TELON DESIGN FACILITY MAIN MENU *****
**********************************************************************
COMMAND ==> __________________________________________________________________

FUNCTION: 2_

      1 —— USER PROFILE MAINTENANCE
      2 —— DATA ADMINISTRATION
      3 —— PANEL SPECIFICATION
      4 —— ONLINE PROGRAM DEFINITION
      5 —— BATCH PROGRAM DEFINITION
      6 —— PROTOTYPING FACILITY
      U —— UTILITIES
      X —— EXIT
```

From the Data Administration menu, type **UP** in the FUNCTION field, **DL** in the ITEM field, the DBD name in the NAME field (in this case, **TRGDBDV1**), and press Enter. CA Telon displays the Update DBD screen.



```
 DATA ADMINISTRATION MENU ************
**********************************************************************
 COMMAND ==> __________________________________________________________________

 FUNCTION:UP   CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:      DL FG-FILE GRP      PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD      TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ   FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:   TRGDBDV1________________________        (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
 DESC:   ________________________________________


 BASE:   ______________________________          (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                         (USED ONLY FOR FUNCTION: CR)
 RDBMS:          ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

From the Update DBD screen, type **U** in the left-hand column of the segment you want to update (in this case, TRGEMPL) and press Enter.

```
TRGDBDV1 UPDATE DBD ****************
**********************************************************************
COMMAND ==> ______________________________________________________ PAGE 01

ACCESS HIDAM,VSAM       RMNAME

SEQ  TYPE    NAME   PARENT/DEVICE MAX LTH SEGMENT KEY LENGTH  START
001 DATASET TRGDBDV1  3390____    _____  ________  _____  _____
U 2 SEGM___ TRGEMPL_  0_______     600  TRGEMKEY    6     1
003 LCHILD_ TRGINDX_  TRGDBDVX     600  TRGEMKEY    6     1
004 SEGM___ TRGTASK_  TRGEMPL_     600  TRGTAKEY    8     1
005 SEGM___ TRGTIME_  TRGTASK_     600  TRGTIKEY    2     1
```

CA Telon redisplays the Update DBD screen for the requested segment. Type **U** in the DLIDSC field of the Update DBD Segment Defaults screen and press Enter to update the DL/I DSC.

```
UPDATE DBD: TRGDBDV1 SEGM: TRGEMPL
**********************************************************************
COMMAND ==> ____________________________________________________________________
OPTIONS ==> SEARCH FIELDS _   PCB PARMS +  DLIDSC U

GENERAL: LABEL  ________
  *   COPY  TRGEMPL_  COPYLV1 _ COPYLBL ______________________________

**  DLIDSC  SEGMENT  CMND  IMSKEY     OP     KEY
01 **DFLT** TRGEMPL  *--- TRGEMKEY =_ XFER-EMPL-ID__________________ _
```

From the screen that displays (the Update DLIDSCs for Segment Member screen), you can type a **U** in the left-hand column of the DL/I DSC segment you want to update (in this case, the GREATER DL/I DSC segment) and press Enter.

```
EDIT ---- UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000004 COL 01
COMMAND ===>                                                       SCROLL
===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL


****** DLIDSC  USECNT   CMND    IMSKEY  OP               KEY               MORE
****** *********** *************       **********       ***********      ****
       ******************************************   *
000001                  **DFLT**        *--- TRGEMKEY    =       XFER-EMPL-ID
U 0002 GREATER  000008  *--- TRGEMKEY    >               TPI-ID
000003 GREATERP IMPORT  *--- TRGEMKEY    >       TPI_ID
000004 GTEQ     000003  *--- TRGEMKEY    >=      TPI-ID
****** *********** *************       **********       ***********      ****
       ******************************************   *
```

CA Telon displays the Create/Update SSAs screen. This screen is used to create or update SSA definitions. It allows you to specify default SSA command information for a segment in a DL/I database. CA Telon uses this information to create DLIDSC statements at export time. Press PF3 to return to the Update SSAs screen.

```
UPDATE SSA/COMMAND FOR DL/I SEGMENT *
*******************************************************************
COMMAND ==> ________________________________________________________________
DBD: TRGDBDV1  SEGMENT: TRGEMPL   DLIDSC: GREATER


GENERAL: KEYFEED ____________________________
  *   CMDCODE *--- -OR- PATH  _ (Y/N)  CURRENT _ (Y/N)  OPTION _ (F/L)
  *            CONCATK _ (Y/N)  PARENTG _ (Y/N)  LOCKED _ (Y/N)
EX DL/I: VARLTH _ (Y/N)   OFFSET ___

WHERE/BOOLEAN SSA:                                                   BOOL
U IMSKEY   OP KEY
       OP
_ TRGEMKEY >_ TPI-ID___________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
_ ________ __ _________________________________________________________ ___
```

From the Update SSAs screen, press PF3 twice to return to the Update DBD Segment Defaults screen.

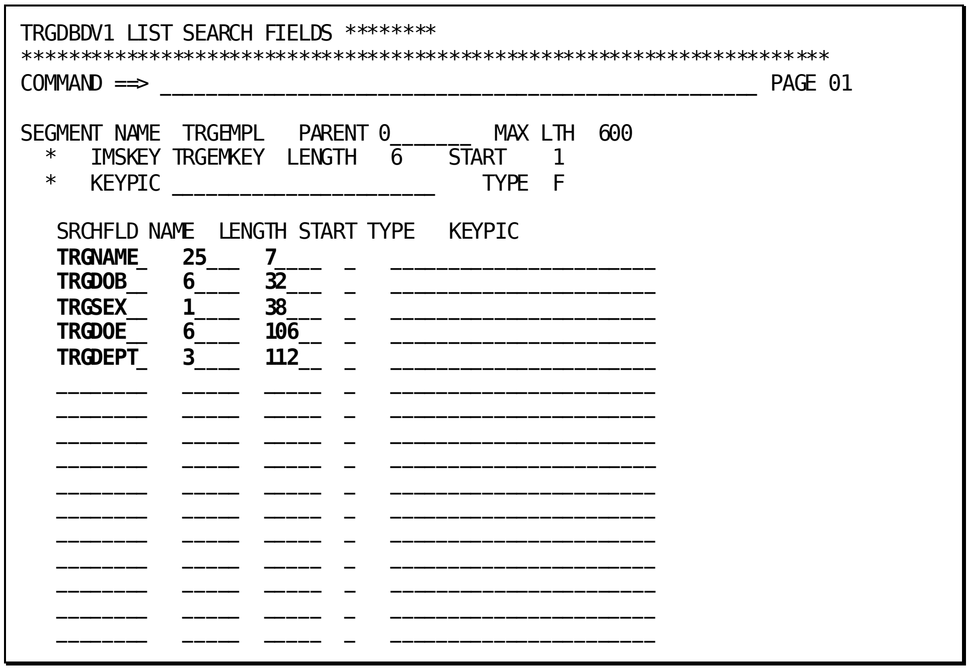
This time on the Update DBD Segment Defaults screen, instead of updating the DLIDSC field, place a **U** in the SEARCH FIELDS field to update search fields.

After you press Enter, the List Search Fields screen displays.

```
UPDATE DBD: TRGDBDV1 SEGM: TRGEMPL  ********************************************
END PROCESSING PERFORMED
COMMAND ==> ________________________________________________________________
OPTIONS ==> SEARCH FIELDS U  PCB PARMS +  DLIDSC _

GENERAL: LABEL  ________
  *   COPY  TRGEMPL_  COPYLV1 _ COPYLBL ____________________________

**  DLIDSC  SEGMENT  CMND  IMSKEY  OP    KEY
01 **DFLT** TRGEMPL  *--- TRGEMKEY =_ XFER-EMPL-ID________________ _
```

The List Search Fields screen enables you to maintain IMS database segment search fields. The following example shows the input of only five of the SSA field names. Make sure all are typed in.

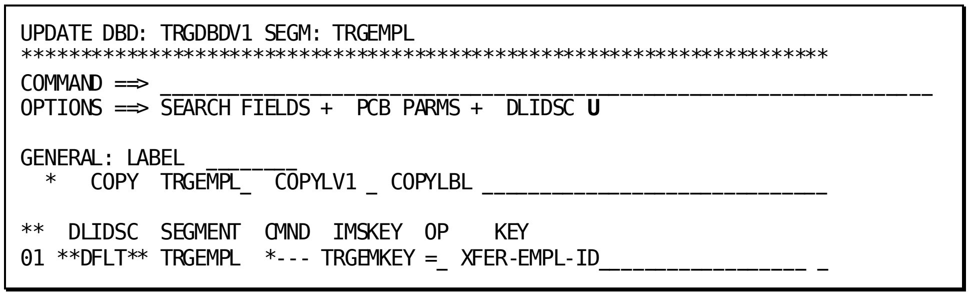
After you finish the input of the SSA field names and data, press PF3 to save the information and you return to the Update DBD Segment Defaults screen.



```
TRGDBDV1 LIST SEARCH FIELDS ********
**********************************************************************
COMMAND ==> ___________________________________________________ PAGE 01

SEGMENT NAME  TRGEMPL   PARENT 0_______  MAX LTH  600
  *   IMSKEY TRGEMKEY  LENGTH   6    START    1
  *   KEYPIC ______________________     TYPE  F

   SRCHFLD NAME  LENGTH START TYPE   KEYPIC
   TRGNAME_   25___  7____  _   _______________________
   TRGDOB__   6____  32___  _   _______________________
   TRGSEX__   1____  38___  _   _______________________
   TRGDOE__   6____  106__  _   _______________________
   TRGDEPT_   3____  112__  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
   ________   _____  _____  _   _______________________
```

From the Update DBD Segment Default screen, you can again update the TRGEMPL segment by typing **U** in the DLIDSC field and pressing Enter.

CA Telon displays the Update SSAs screen.



```
UPDATE DBD: TRGDBDV1 SEGM: TRGEMPL
**********************************************************************
COMMAND ==> __________________________________________________________________
OPTIONS ==> SEARCH FIELDS +  PCB PARMS +  DLIDSC U

GENERAL: LABEL  ________
  *   COPY  TRGEMPL_  COPYLV1 _ COPYLBL ______________________________

**  DLIDSC  SEGMENT  CMND  IMSKEY  OP    KEY
01 **DFLT** TRGEMPL  *--- TRGEMKEY =_ XFER-EMPL-ID__________________ _
```

From the Update SSAs screen, you can insert a line after the last SSA on the list by typing **I** in the left-hand column of that SSA and pressing Enter.

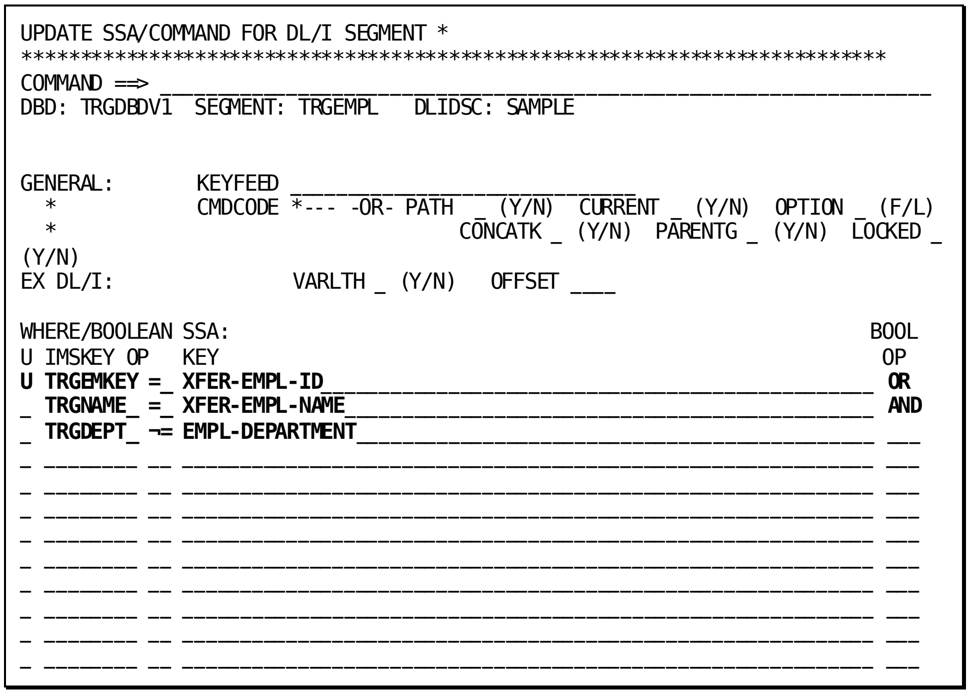
```
EDIT ---- UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000004 COL 01
COMMAND ===>                                                         SCROLL
===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL
****** DLIDSC  USECNT   CMND    IMSKEY  OP              KEY             MORE
****** *********** *************       **********       ***********     ****
       *****************************************   *
000001                  **DFLT**        *--- TRGEMKEY   =       XFER-EMPL-ID
000002 GREATER  000008  *--- TRGEMKEY   >               TPI-ID
000003 GREATERP IMPORT  *--- TRGEMKEY   >       TPI_ID
I 0004 GTEQ     000003  *--- TRGEMKEY   >=      TPI-ID
****** *********** *************       **********       ***********     ****
       *****************************************   *
```

You can process on the line by typing **U** in the left-hand column and the DLIDSC name in that column (in this case, **SAMPLE**) and pressing Enter. CA Telon displays the Create/Update SSAs screen.

```
EDIT ---- UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000004 COL 01
COMMAND ===>                                                         SCROLL
===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL

****** DLIDSC  USECNT   CMND    IMSKEY  OP              KEY             MORE
****** *********** *************       **********       ***********     ****
       *****************************************   *
000001                  **DFLT**        *--- TRGEMKEY   =       XFER-EMPL-ID
000002 GREATER  000008  *--- TRGEMKEY   >               TPI-ID
000003 GREATERP IMPORT  *--- TRGEMKEY   >       TPI_ID
000004 GTEQ     000003  *--- TRGEMKEY   >=      TPI-ID
U '''' SAMPLE
****** *********** *************       **********       ***********     ****
       *****************************************   *
```

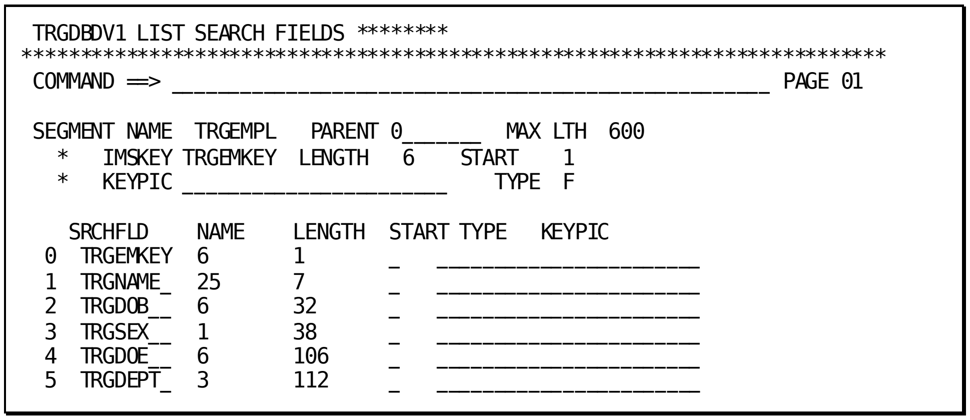
From the Create/Update SSAs screen you can create or update SSA definitions. Notice how this example has two Boolean operations (OR and AND) on SSAs.

To update a Boolean SSA, type a **U** in the left-hand column of the key you want to update and press Enter.



```
UPDATE SSA/COMMAND FOR DL/I SEGMENT *
***********************************************************************
COMMAND ==> ____________________________________________________________
DBD: TRGDBDV1  SEGMENT: TRGEMPL   DLIDSC: SAMPLE


GENERAL:       KEYFEED ____________________________
  *            CMDCODE *--- -OR- PATH  _ (Y/N)  CURRENT _ (Y/N)  OPTION _ (F/L)
  *                                 CONCATK _ (Y/N)  PARENTG _ (Y/N)  LOCKED _
(Y/N)
EX DL/I:               VARLTH _ (Y/N)   OFFSET ___

WHERE/BOOLEAN SSA:                                                BOOL
U IMSKEY OP   KEY                                                  OP
U TRGEMKEY =_ XFER-EMPL-ID____________________________________________ OR
_ TRGNAME_ =_ XFER-EMPL-NAME__________________________________________ AND
_ TRGDEPT_ ¬= EMPL-DEPARTMENT_________________________________________ ___
_ ________ __ ________________________________________________________ ___
_ ________ __ ________________________________________________________ ___
_ ________ __ ________________________________________________________ ___
_ ________ __ ________________________________________________________ ___
_ ________ __ ________________________________________________________ ___
_ ________ __ ________________________________________________________ ___
_ ________ __ ________________________________________________________ ___
_ ________ __ ________________________________________________________ ___
_ ________ __ ________________________________________________________ ___
```

CA Telon displays the List Search Fields screen. Press PF3 to save any edited information on this screen and you return to the Update SSAs screen.



```
 TRGDBDV1 LIST SEARCH FIELDS ********
***********************************************************************
 COMMAND ==> ________________________________________________ PAGE 01

 SEGMENT NAME  TRGEMPL   PARENT 0______  MAX LTH  600
   *   IMSKEY TRGEMKEY  LENGTH   6    START    1
   *   KEYPIC _____________________     TYPE  F

    SRCHFLD     NAME    LENGTH  START TYPE   KEYPIC
  0  TRGEMKEY  6        1         _   _____________________
  1  TRGNAME_  25       7         _   _____________________
  2  TRGDOB__  6        32        _   _____________________
  3  TRGSEX__  1        38        _   _____________________
  4  TRGDOE__  6        106       _   _____________________
  5  TRGDEPT_  3        112       _   _____________________
```

From the Update SSAs screen, press PF3 to access the Update DBD Segment Defaults screen.

```
EDIT ---- UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000005 COL 01
COMMAND ===>                               SCROLL ===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL
****** DLIDSC  USECNT   CMND    IMSKEY  OP              KEY               MORE
****** *********** *************       **********       ***********      ****
       *****************************************   *
000001                 **DFLT**        *--- TRGEMKEY    =       XFER-EMPL-ID
000002 GREATER  000008 *--- TRGEMKEY    >               TPI-ID
000003 GREATERP IMPORT *--- TRGEMKEY    >       TPI_ID
000004 GTEQ    000003  *--- TRGEMKEY    >=      TPI-ID
000005 SAMPLE          **DA**  *--- TRGEMKEY    >=      XFER-EMPL-ID
                                                +
****** *********** *************       **********       ***********      ****
       *****************************************   *
```

Notice on the Update SSAs screen how CA Telon displays **DA** in the USECNT field of the SAMPLE DLIDSC to indicate that information was brought in using Data Administration. Notice also how you can only see the first line of the Boolean operation (TRGEMKEY = XFER-EMPL-ID). The + in the MORE column indicates there is more information than that which is shown on this screen.

From the Update DLIDSCS screen, press PF3 to return to the Update DBD Segment Defaults screen.

```
UPDATE DBD: TRGDBDV1 SEGM: TRGEMPL  ************************* END PROCESSING
PERFORMED
COMMAND ==> ________________________________________________________________
OPTIONS ==> SEARCH FIELDS +  PCB PARMS +  DLIDSC _

GENERAL: LABEL  ________
  *   COPY  TRGEMPL_  COPYLV1 _ COPYLBL ______________________________

**  DLIDSC  SEGMENT  CMND  IMSKEY   OP    KEY
01 **DFLT** TRGEMPL  *--- TRGEMKEY =_ XFER-EMPL-ID__________________ _
```

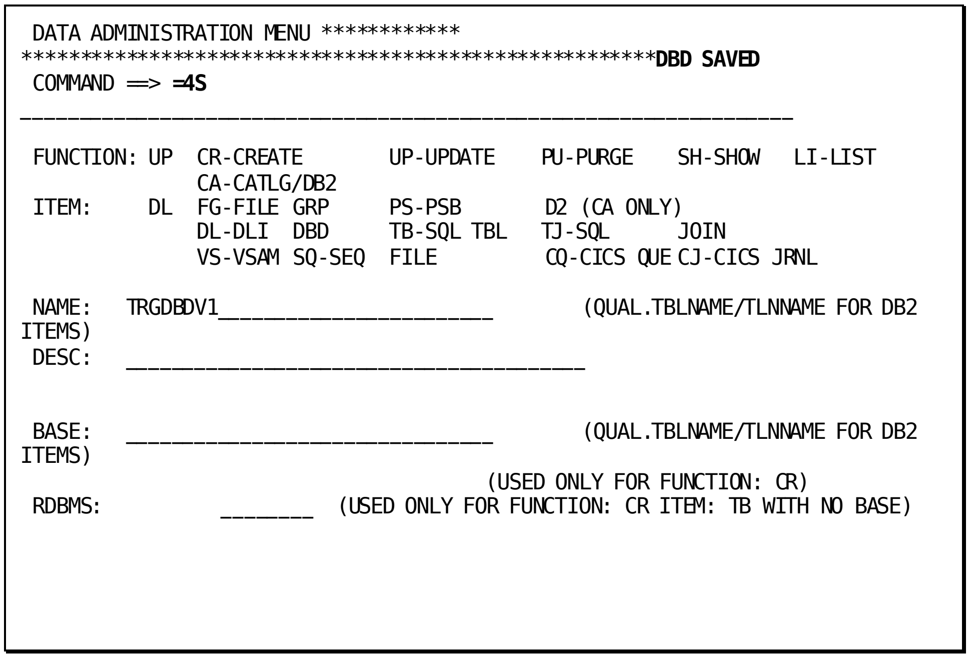
From the Update DBD Segment Defaults screen, press PF3 to access the Update DBD screen. Notice how the message **SEGMENT DEFAULT INFORMATION UPDATED** appears in the upper right-hand corner of the Update DBD screen.

```
TRGDBDV1 UPDATE DBD **************** **************************************
SEGMENT DEFAULT INFORMATION UPDATED
COMMAND ==> _____________________________________________________ PAGE 01

ACCESS HIDAM,VSAM       RMNAME

SEQ  TYPE    NAME  PARENT/DEVICE MAX LTH SEGMENT KEY LENGTH  START
001 DATASET TRGDBDV1  3390____    _____ ________  _____ _____
002 SEGM___ TRGEMPL_  0_______     600  TRGEMKEY    6    1
003 LCHILD_ TRGINDX_  TRGDBDVX     600  TRGEMKEY    6    1
004 SEGM___ TRGTASK_  TRGEMPL_     600  TRGTAKEY    8    1
005 SEGM___ TRGTIME_  TRGTASK_     600  TRGTIKEY    2    1
```

From the Update DBD screen, press PF3 to return to the Data Administration menu. Notice how the message **DBD SAVED** appears in the upper right-hand corner.

```
 DATA ADMINISTRATION MENU ************
*****************************************************************DBD SAVED
 COMMAND ==> ____________________________________________________________________

 FUNCTION: UP  CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:     DL  FG-FILE GRP      PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD      TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ   FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:   TRGDBDV1________________________        (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
 DESC:   ________________________________________


 BASE:   ________________________________        (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                         (USED ONLY FOR FUNCTION: CR)
 RDBMS:          ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

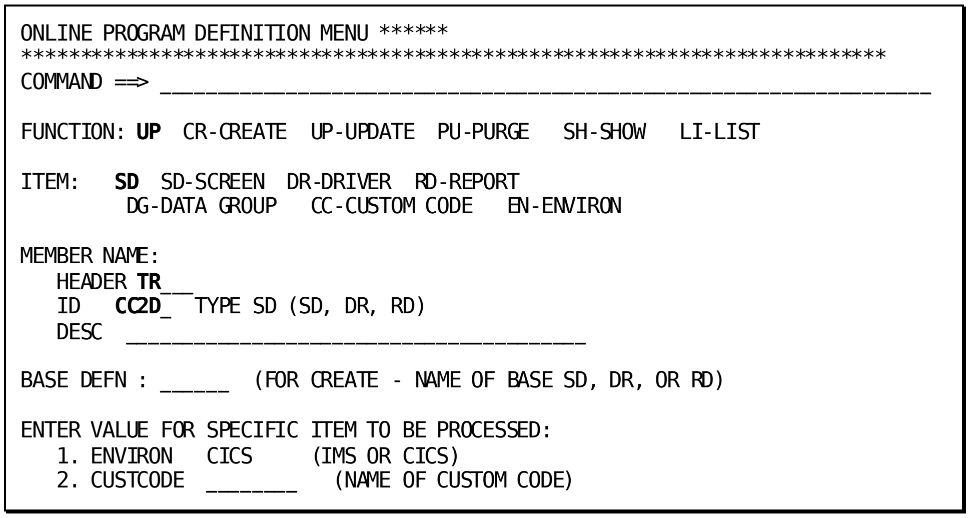
Type **=4S** on the COMMAND line of the Data Administration menu to transfer to the Online Program Definition menu to update the screen definition.



```
 DATA ADMINISTRATION MENU ************
*******************************************************DBD SAVED
 COMMAND ==> =4S
___________________________________________________________________

 FUNCTION: UP  CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:     DL  FG-FILE GRP      PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD      TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ   FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:    TRGDBDV1________________________       (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
 DESC:    ________________________________________


 BASE:    ________________________________       (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                         (USED ONLY FOR FUNCTION: CR)
 RDBMS:          ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

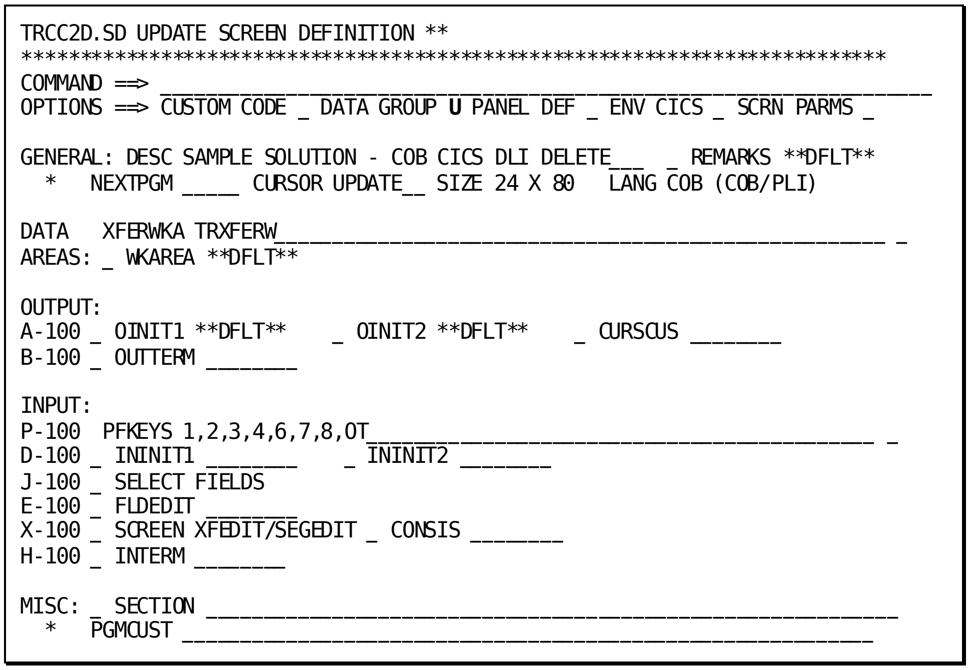
How a DSC is Used in a Data Group

To update the screen definition, type **UP** in the FUNCTION field, **SD** in the ITEM field, and the appropriate HEADER and ID (in this case, **TR** and **CC2D**) on the Online Program Definition menu and press Enter.



```
ONLINE PROGRAM DEFINITION MENU ******
***************************************************************************
COMMAND ==> __________________________________________________________________

FUNCTION: UP  CR-CREATE  UP-UPDATE  PU-PURGE   SH-SHOW   LI-LIST

ITEM:    SD  SD-SCREEN  DR-DRIVER  RD-REPORT
          DG-DATA GROUP   CC-CUSTOM CODE   EN-ENVIRON

MEMBER NAME:
   HEADER TR__
   ID    CC2D_  TYPE SD (SD, DR, RD)
   DESC   ________________________________________

BASE DEFN : ______  (FOR CREATE - NAME OF BASE SD, DR, OR RD)

ENTER VALUE FOR SPECIFIC ITEM TO BE PROCESSED:
   1. ENVIRON   CICS     (IMS OR CICS)
   2. CUSTCODE  ________   (NAME OF CUSTOM CODE)
```

CA Telon displays the Update Screen Definition screen. From this screen, type **U** in the DATA GROUP field and press Enter to update the data group.

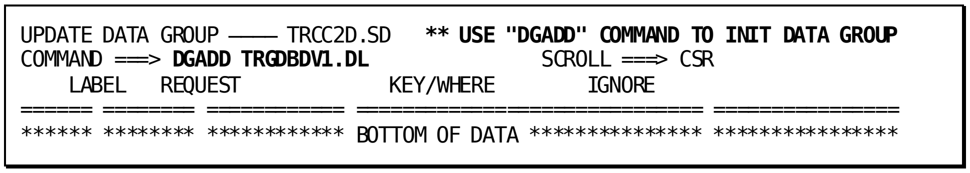


```
TRCC2D.SD UPDATE SCREEN DEFINITION **
*************************************************************************
COMMAND ==> ____________________________________________________________________
OPTIONS ==> CUSTOM CODE _ DATA GROUP U PANEL DEF _ ENV CICS _ SCRN PARMS _

GENERAL: DESC SAMPLE SOLUTION - COB CICS DLI DELETE___  _ REMARKS **DFLT**
  *   NEXTPGM _____ CURSOR UPDATE__ SIZE 24 X 80   LANG COB (COB/PLI)

DATA   XFERWKA TRXFERW_____________________________________________________ _
AREAS: _ WKAREA **DFLT**

OUTPUT:
A-100 _ OINIT1 **DFLT**    _ OINIT2 **DFLT**    _ CURSCUS ________
B-100 _ OUTTERM ________

INPUT:
P-100  PFKEYS 1,2,3,4,6,7,8,OT_________________________________________ _
D-100 _ ININIT1 ________    _ ININIT2 ________
J-100 _ SELECT FIELDS
E-100 _ FLDEDIT ________
X-100 _ SCREEN XFEDIT/SEGEDIT _ CONSIS ________
H-100 _ INTERM ________

MISC: _ SECTION _____________________________________________________________
  *   PGMCUST _____________________________________________________________
```

CA Telon displays the Update Data Group screen with a prompt in the upper right-hand corner of the screen to remind you to use the DGADD command.

Type the appropriate DGADD command in the command line (in this case, **DGADD TRGDBDV1.DL**) and press Enter. CA Telon redisplays the screen with the requested information.

For a complete list of the DGADD commands available on this screen, refer to the *Design* Facility *Reference Guide*.



```
UPDATE DATA GROUP ---- TRCC2D.SD   ** USE "DGADD" COMMAND TO INIT DATA GROUP
COMMAND ===> DGADD TRGDBDV1.DL                  SCROLL ===> CSR
    LABEL   REQUEST             KEY/WHERE        IGNORE
====== ======== ============ ============================== ================
****** ******** ************ BOTTOM OF DATA *************** ****************
```

From the Update Data Group screen, you can update the data access level by typing **U** in the left-hand column of what you want to update and pressing Enter.

```
UPDATE DATA GROUP ———— TRCC2D.SD            SIZE 000005 COL 01
COMMAND ===>                            SCROLL ===> CSR
    LABEL   REQUEST       KEY/WHERE        IGNORE
====== ======= =========== =========================== ==============
PCB==> TRGDBDV1 TRGDBDV1
U EG=> TRGEMPL @DUMMY    @XFER-EMPL-ID
 SEG=> TRGTASK @DUMMY    @XFER-TASK-KEY
 SEG=> TRGTIME @DUMMY    @XFER-TIME-KEY
****** ******** ************ BOTTOM OF DATA *************** ***************
```

CA Telon displays the Update Database Segment screen.

From the Update Database Segment screen, you can do further updates of a data segment (in this case, TRGEMPL). You can update by typing **U** in the left-hand column of the segment you want to update and press Enter.

```
UPDATE DBD: TRGDBDV1 SEGM: TRGEMPL
******************************************************************************
*
COMMAND ==> ________________________________________________________________
OPTIONS ==> PCB PARMS +

GENERAL: LABEL  TRGEMPL_  USAGE @DUMMY__
  *   COPY  @TRGEMPL_ COPYLV1 @_ COPYLBL @___________________________


** DSCREF SEGMENT CMND  IMSKEY  OP    KEY
U1 **DFLT** TRGEMPL @*--- @TRGEMKEY @=_ @XFER-EMPL-ID________________ _
```

CA Telon displays the Update DLIDSCs screen. The + in the MORE column indicates there is more information than that which is shown on the screen.

From the Update DLIDSCs screen, type **U** in the left-hand column and press Enter. This enables you to see the full Boolean expression.

```
SELECT —— UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000005 COL 01
COMMAND ===>                            SCROLL ===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL

****** DLIDSC  USECNT CMND  IMSKEY  OP    KEY           MORE
****** ******** ****** **** ******** ** ****************************** *
000001 **DFLT**     *--- TRGEMKEY =  XFER-EMPL-ID
000002 GREATER  000008 *--- TRGEMKEY >  TPI-ID
000003 GREATERP IMPORT *--- TRGEMKEY >  TPI_ID
000004 GTEQ   000003 *--- TRGEMKEY >= TPI-ID
U 0005 SAMPLE  **DA** *--- TRGEMKEY =  XFER-EMPL-ID          +
****** ******** ****** **** ******** ** ****************************** *
```

CA Telon displays the Create/Update SSAs screen. This shows the entire Boolean expression for the "SAMPLE" DSC of the TRGEMPL segment.

```
UPDATE SSA/COMMAND FOR DL/I SEGMENT *
****************************************************************************
*
COMMAND ==> ___________________________________________________________________
DBD: TRGDBDV1  SEGMENT: TRGEMPL   DLIDSC: SAMPLE


GENERAL: KEYFEED _____________________________
  *   CMDCODE *--- -OR- PATH  _ (Y/N)  CURRENT _ (Y/N)  OPTION _ (F/L)
  *            CONCATK _ (Y/N)  PARENTG _ (Y/N)  LOCKED _ (Y/N)
EX DL/I: VARLTH _ (Y/N)   OFFSET ____

WHERE/BOOLEAN SSA:                            BOOL
U IMSKEY OP   KEY                          OP
_ TRGEMKEY =_ XFER-EMPL-ID___________________________________________ OR_
_ TRGNAME_ =_ XFER-EMPL-NAME_________________________________________ AND
_ TRGDEPT_ ¬= EMPL-DEPARTMENT________________________________________ ___
_ ________ __ _______________________________________________________ ___
_ ________ __ _______________________________________________________ ___
_ ________ __ _______________________________________________________ ___
_ ________ __ _______________________________________________________ ___
_ ________ __ _______________________________________________________ ___
_ ________ __ _______________________________________________________ ___
_ ________ __ _______________________________________________________ ___
_ ________ __ _______________________________________________________ ___
_ ________ __ _______________________________________________________ ___
```

Press PF3 twice to return to the Update Data Group screen.

From the Update Data Group screen, insert a line by typing **I** in the left-hand column of the segment you want to look at. In this case, you want to do an AUTOEXEC OUTREAD of the TRGEMPL segment.

```
UPDATE DATA GROUP ----- TRCC2D.SD            SIZE 000005 COL 01
COMMAND ===>                           SCROLL ===> CSR
    LABEL   REQUEST       KEY/WHERE        IGNORE
====== ======== =========== ============================ ===============
PCB==> TRGDBDV1 TRGDBDV1
I EG=> TRGEMPL @DUMMY    @XFER-EMPL-ID
 SEG=> TRGTASK @DUMMY    @XFER-TASK-KEY
 SEG=> TRGTIME @DUMMY    @XFER-TIME-KEY
****** ******** ************ BOTTOM OF DATA *************** ****************
```

On the inserted line type **V** as the line command, **AUTOEXEC** as the LABEL, **OUTREAD** as the request and **@** on the KEY/WHERE column.

```
UPDATE DATA GROUP ----- TRCC2D.SD             SIZE 000005 COL 01
COMMAND ===>                          SCROLL ===> CSR
    LABEL   REQUEST       KEY/WHERE        IGNORE
====== ======= =========== ========================== ===============
PCB==> TRGDBDV1 TRGDBDV1
 SEG=> TRGEMPL @DUMMY    @XFER-EMPL-ID
V''''' AUTOEXEC OUTREAD   @
 SEG=> TRGTASK @DUMMY    @XFER-TASK-KEY
 SEG=> TRGTIME @DUMMY    @XFER-TIME-KEY
****** ******** ************ BOTTOM OF DATA *************** ****************
```

After you press Enter, CA Telon displays the DL/I Preview screen that shows what the generated code will look like.

```
VIEW ----- TRCC2D.SD *PREVIEW    MEMBER 001 OF 001  SIZE 000022 COL 07
COMMAND ===>                          SCROLL ===> CSR
000001 *  DEFAULT GENERATED USER I/O
000002 *  DEFAULT CALL: AUTOEXEC OUTREAD TRGEMPL *
000003
000004 *  ** SSAS IN STORAGE
000005
000006   05 TRGEMPL-SSA.
000007    10 TRGEMPL-SSA-SEGMENT      PIC X(8) VALUE 'TRGEMPL'.
000008    10 TRGEMPL-SSA-CMDCODE      PIC X(4) VALUE '*---'.
000009    10 TRGEMPL-SSA-LPAREN       PIC X(1) VALUE '('.
000010    10 TRGEMPL-SSA-IMSKEY       PIC X(8) VALUE 'TRGEMKEY'.
000011    10 TRGEMPL-SSA-OPCODE       PIC X(2) VALUE '='.
000012    10 TRGEMPL-SSA-VALUE       PIC X(006).
000013    10 FILLER             PIC X(2) VALUE ')'.
000014
000015 *  ** ASSIGNMENT STATEMENT FOR KEY
000016
000017   MOVE XFER-EMPL-ID TO TRGEMPL-SSA-VALUE.
000018
000019   CALL 'CBLTDLI' USING GU-FUNC
000020              TRGDBDV1-PCB
000021              IOA-TRGEMPL-SEGMENT
000022              TRGEMPL-SSA.
```

Press PF3 to return to the Update Data Group screen.

You can then update the detail for this DL/I data access by typing **U** in the left-hand column of the AUTOEXEC OUTREAD and press Enter.

```
UPDATE DATA GROUP ---- TRCC2D.SD            SIZE 000006 COL 01
COMMAND ==>                         SCROLL ==> CSR
    LABEL   REQUEST       KEY/WHERE       IGNORE
====== ======= =========== ============================ ===============
PCB==> TRGDBDV1 TRGDBDV1
 SEG=> TRGEMPL @DEFINE   @XFER-EMPL-ID
U 0001 AUTOEXEC OUTREAD   @
 SEG=> TRGTASK @DUMMY    @XFER-TASK-KEY
 SEG=> TRGTIME @DUMMY    @XFER-TIME-KEY
****** ******** ************ BOTTOM OF DATA ************** ***************
```

CA Telon displays the DL/I Detail Data Access screen. You can type a **U** in the left-hand column of a SEGMENT (in this case, TRGEMPL) and press Enter to request a list of SSAs for the segment.

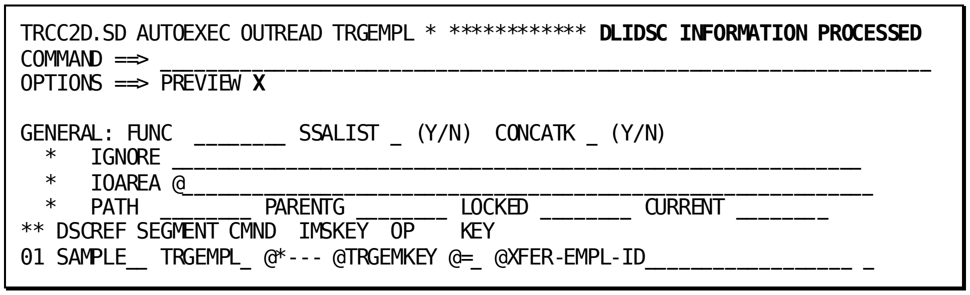
```
TRCC2D.SD AUTOEXEC OUTREAD TRGEMPL *
*****************************************************************************
*
COMMAND ==> ________________________________________________________________
OPTIONS ==> PREVIEW _

GENERAL: FUNC  _______  SSALIST _ (Y/N)  CONCATK _ (Y/N)
  *   IGNORE _________________________________________________________
  *   IOAREA @_________________________________________________________
  *   PATH  _______  PARENTG _______  LOCKED _______  CURRENT _______
** DSCREF SEGMENT CMND  IMSKEY  OP    KEY
U1 **DFLT** TRGEMPL_ @*--- @TRGEMKEY @=_ @XFER-EMPL-ID_________________ _
```

CA Telon displays the Update SSAs screen. From the Update SSAs screen, type **S** in the left-hand column to select an alternate DSC (in this case, SAMPLE) as shown in the example below. Then press PF3 to save this information.

```
SELECT -- UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000005 COL 01
COMMAND ==>                         SCROLL ==> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL

****** DLIDSC  USECNT CMND  IMSKEY  OP    KEY           MORE
****** ******** ****** **** ******** ** ***************************** *
000001 **DFLT**     *--- TRGEMKEY =  XFER-EMPL-ID
000002 GREATER  000008 *--- TRGEMKEY >  TPI-ID
000003 GREATERP IMPORT *--- TRGEMKEY >  TPI_ID
000004 GTEQ   000003 *--- TRGEMKEY >= TPI-ID
S 0005 SAMPLE  **DA** *--- TRGEMKEY =  XFER-EMPL-ID          +
****** ******** ****** **** ******** ** ***************************** *
```

CA Telon displays the DL/I Detail Data Access screen. Notice how CA Telon displays the message **DLIDSC INFORMATION PROCESSED**. Notice also that the DSCNAME was changed to "SAMPLE" which is the name that was just selected.

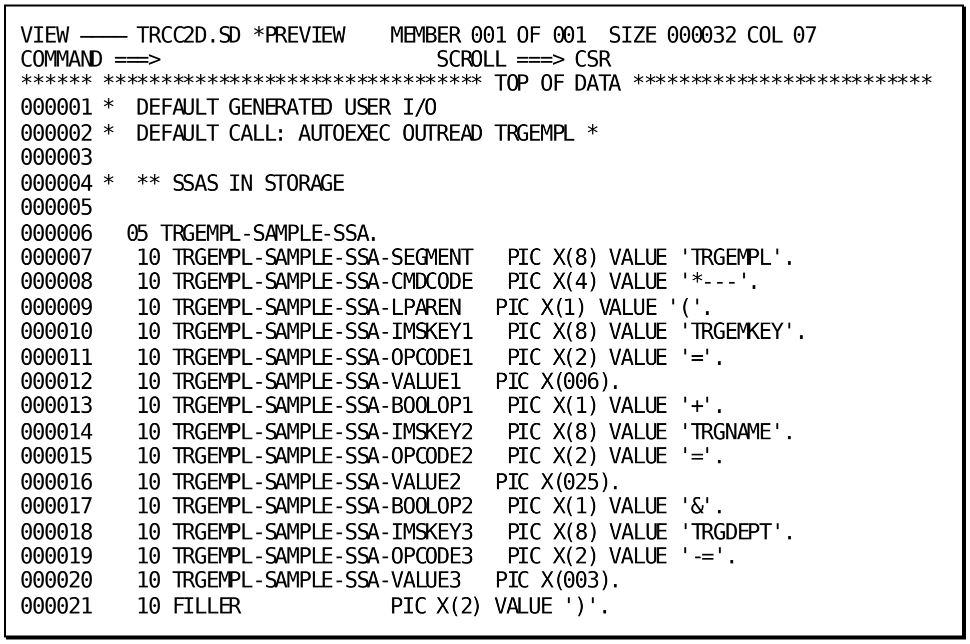
From the DL/I Detail Data Access screen, type a non-blank character in the OPTIONS field after the PREVIEW literal and press Enter to preview the segment (in this case, TRGEMPL).



```
TRCC2D.SD AUTOEXEC OUTREAD TRGEMPL * ************ DLIDSC INFORMATION PROCESSED
COMMAND ==> ________________________________________________________________
OPTIONS ==> PREVIEW X

GENERAL: FUNC  _______ SSALIST _ (Y/N)  CONCATK _ (Y/N)
  *   IGNORE _______________________________________________________
  *   IOAREA @______________________________________________________
  *   PATH  _______ PARENTG _______ LOCKED _______ CURRENT _______
** DSCREF SEGMENT CMND  IMSKEY  OP    KEY
01 SAMPLE__ TRGEMPL_ @*--- @TRGEMKEY @=_ @XFER-EMPL-ID________________ _
```

CA Telon displays the DL/I Preview screen. This gives a preview of what the generated code will look like. Notice how the Boolean operator OR (shown by the + sign) and the Boolean operator AND (shown by the & symbol) are shown.

Press PF8 to page down to the next screen.



```
VIEW ---- TRCC2D.SD *PREVIEW    MEMBER 001 OF 001  SIZE 000032 COL 07
COMMAND ===>                           SCROLL ===> CSR
****** ******************************** TOP OF DATA *************************
000001 *  DEFAULT GENERATED USER I/O
000002 *  DEFAULT CALL: AUTOEXEC OUTREAD TRGEMPL *
000003
000004 *  ** SSAS IN STORAGE
000005
000006   05 TRGEMPL-SAMPLE-SSA.
000007    10 TRGEMPL-SAMPLE-SSA-SEGMENT   PIC X(8) VALUE 'TRGEMPL'.
000008    10 TRGEMPL-SAMPLE-SSA-CMDCODE   PIC X(4) VALUE '*---'.
000009    10 TRGEMPL-SAMPLE-SSA-LPAREN   PIC X(1) VALUE '('.
000010    10 TRGEMPL-SAMPLE-SSA-IMSKEY1   PIC X(8) VALUE 'TRGEMKEY'.
000011    10 TRGEMPL-SAMPLE-SSA-OPCODE1   PIC X(2) VALUE '='.
000012    10 TRGEMPL-SAMPLE-SSA-VALUE1   PIC X(006).
000013    10 TRGEMPL-SAMPLE-SSA-BOOLOP1   PIC X(1) VALUE '+'.
000014    10 TRGEMPL-SAMPLE-SSA-IMSKEY2   PIC X(8) VALUE 'TRGNAME'.
000015    10 TRGEMPL-SAMPLE-SSA-OPCODE2   PIC X(2) VALUE '='.
000016    10 TRGEMPL-SAMPLE-SSA-VALUE2   PIC X(025).
000017    10 TRGEMPL-SAMPLE-SSA-BOOLOP2   PIC X(1) VALUE '&'.
000018    10 TRGEMPL-SAMPLE-SSA-IMSKEY3   PIC X(8) VALUE 'TRGDEPT'.
000019    10 TRGEMPL-SAMPLE-SSA-OPCODE3   PIC X(2) VALUE '-='.
000020    10 TRGEMPL-SAMPLE-SSA-VALUE3   PIC X(003).
000021    10 FILLER             PIC X(2) VALUE ')'.
```

The following example shows the rest of the DL/I Preview screen for the generated code of the TRGEMPL segment.

```
VIEW ---- TRCC2D.SD *PREVIEW    MEMBER 001 OF 001  SIZE 000032 COL 07
COMMAND ===>                           SCROLL ===> CSR
000022
000023 *  ** ASSIGNMENT STATEMENT FOR KEY
000024
000025   MOVE XFER-EMPL-ID TO TRGEMPL-SAMPLE-SSA-VALUE1.
000026   MOVE XFER-EMPL-NAME TO TRGEMPL-SAMPLE-SSA-VALUE2.
000027   MOVE EMPL-DEPARTMENT TO TRGEMPL-SAMPLE-SSA-VALUE3.
000028
000029   CALL 'CBLTDLI' USING GU-FUNC
000030                TRGDBDV1-PCB
000031                IOA-TRGEMPL-SEGMENT
000032                TRGEMPL-SAMPLE-SSA.
****** ******************************** BOTTOM OF DATA **********************
```

Press PF3 to return to the Update Data Group screen.

Type a **U** in the left-hand column of the AUTOEXEC and press Enter to update it.

```
UPDATE DATA GROUP ----- TRCC2D.SD            SIZE 000006 COL 01
COMMAND ===>                           SCROLL ===> CSR
    LABEL      REQUEST       KEY/WHERE        IGNORE
====== ======== =========== ============================= ===============
PCB==> TRGDBDV1 TRGDBDV1
 SEG=> TRGEMPL @DEFINE    @XFER-EMPL-ID
U 0001 AUTOEXEC OUTREAD    @-DSC                      +
 SEG=> TRGTASK @DUMMY     @XFER-TASK-KEY
 SEG=> TRGTIME @DUMMY     @XFER-TIME-KEY
****** ******** ************ BOTTOM OF DATA *************** ****************
```

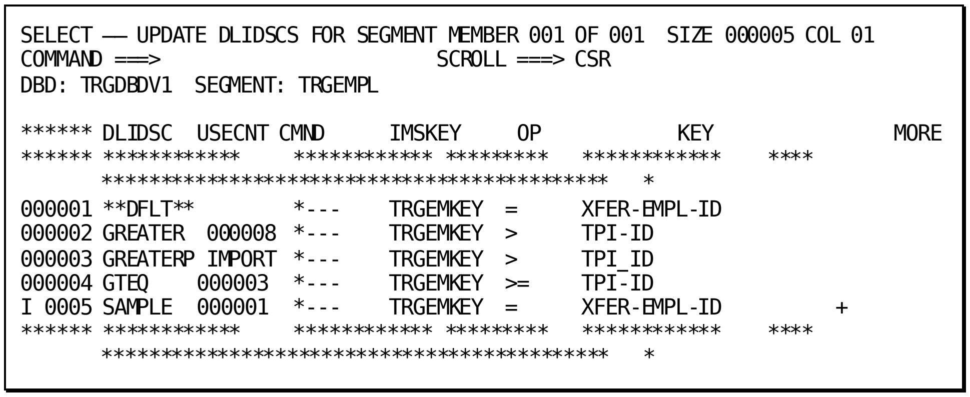
CA Telon displays the DL/I Detail Data Access screen for the AUTOEXEC. Type **U** in the left-hand column of the segment you want (in this case, TRGEMPL) on the DL/I Detail Data Access screen.

```
TRCC2D.SD AUTOEXEC OUTREAD TRGEMPL * ****************************************
COMMAND ==> _________________________________________________________________
OPTIONS ==> PREVIEW _

GENERAL: FUNC ________ SSALIST _ (Y/N)  CONCATK _ (Y/N)
  *   IGNORE ___________________________________________________________
  *   IOAREA @___________________________________________________________
  *   PATH ________ PARENTG ________ LOCKED ________ CURRENT ________
** DSCREF SEGMENT CMND  IMSKEY  OP    KEY
U1 SAMPLE__ TRGEMPL_ @*--- @TRGEMKEY @=_ @XFER-EMPL-ID_________________ _
```

After you press Enter, CA Telon displays the Update SSAs screen.

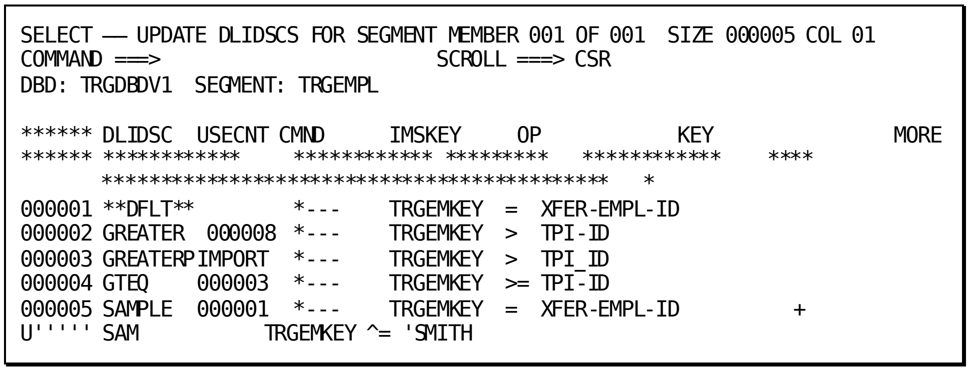
You can create a new DSC in your data group as you continue processing. Do this from the Update SSAs screen by typing **I** in the left-hand column of the DSC already created (in this case, SAMPLE).

press Enter to insert a line.



```
SELECT —— UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000005 COL 01
COMMAND ===>                            SCROLL ===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL

****** DLIDSC  USECNT CMND      IMSKEY     OP            KEY               MORE
****** ************     ************ *********    ************    ****
       ********************************************    *
000001 **DFLT**         *---     TRGEMKEY  =     XFER-EMPL-ID
000002 GREATER  000008  *---     TRGEMKEY  >     TPI-ID
000003 GREATERP IMPORT  *---     TRGEMKEY  >     TPI_ID
000004 GTEQ     000003  *---     TRGEMKEY  >=    TPI-ID
I 0005 SAMPLE   000001  *---     TRGEMKEY  =     XFER-EMPL-ID          +
****** ************     ************ *********    ************    ****
       ********************************************    *
```

On the Update SSAs screen, type a new DSC name (in this case, **SAM**), type an operation (in this case, **TRGEMKEY &caret.= 'SMITH'**), type a **U** in the left-hand column and press Enter.



```
SELECT —— UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000005 COL 01
COMMAND ===>                            SCROLL ===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL

****** DLIDSC  USECNT CMND      IMSKEY     OP            KEY               MORE
****** ************     ************ *********    ************    ****
       ********************************************    *
000001 **DFLT**         *---     TRGEMKEY  =  XFER-EMPL-ID
000002 GREATER  000008  *---     TRGEMKEY  >  TPI-ID
000003 GREATERPIMPORT   *---     TRGEMKEY  >  TPI_ID
000004 GTEQ     000003  *---     TRGEMKEY  >= TPI-ID
000005 SAMPLE   000001  *---     TRGEMKEY  =  XFER-EMPL-ID          +
U''''' SAM           TRGEMKEY ^= 'SMITH
```

CA Telon displays the Create/Update SSAs screen with the specified operation display. Type a **U** in the left-hand column to update this operation, and press Enter.

```
UPDATE SSA/COMMAND FOR DL/I SEGMENT * ****************************************
COMMAND ==> _________________________________________________________________
DBD: TRGDBDV1  SEGMENT: TRGEMPL   DLIDSC: SAM


GENERAL: KEYFEED _____________________________
  *   CMDCODE *--- -OR- PATH  _ (Y/N)  CURRENT _ (Y/N)  OPTION _ (F/L)
  *            CONCATK _ (Y/N)  PARENTG _ (Y/N)  LOCKED _ (Y/N)
EX DL/I: VARLTH _ (Y/N)   OFFSET ___

WHERE/BOOLEAN SSA:                           BOOL
U IMSKEY OP   KEY                          OP
U TRGEMKEY &caret.= 'SMITH'___________________________________________________
___
_ ________ __ ___________________________________________________________ ___
_ ________ __ ___________________________________________________________ ___
_ ________ __ ___________________________________________________________ ___
_ ________ __ ___________________________________________________________ ___
```

CA Telon displays the List Search Fields screen.

From the List Search Fields screen, press PF3 to return to the Create/Update SSAs screen.

```
TRGDBDV1 LIST SEARCH FIELDS ******** ****************************************
COMMAND ==> ___________________________________________________ PAGE 01

SEGMENT NAME  TRGEMPL   PARENT 0_______  MAX LTH  600
  *   IMSKEY TRGEMKEY  LENGTH   6    START    1
  *   KEYPIC ______________________     TYPE  F

   SRCHFLD NAME  LENGTH START TYPE   KEYPIC
 0  TRGEMKEY     6    1  _   ______________________
 1  TRGNAME_     25    7  _   ______________________
 2  TRGDOB__     6   32  _   ______________________
 3  TRGSEX__     1   38  _   ______________________
 4  TRGDOE__     6   106  _   ______________________
 5  TRGDEPT_     3   112  _   ______________________
```

On the Create/Update SSAs screen, you can edit the operation to make it a Boolean operation. The following example shows the typing in of **AND** and **TRGEMKEY &caret.= "JONES"**. press Enter after you have made edited changes.

CA Telon returns you to the Update SSAs screen.

```
UPDATE SSA/COMMAND FOR DL/I SEGMENT * ****************************************
COMMAND ==> __________________________________________________________________
DBD: TRGDBDV1  SEGMENT: TRGEMPL   DLIDSC: SAM


GENERAL: KEYFEED ______________________________
  *   CMDCODE *--- -OR- PATH  _ (Y/N)  CURRENT _ (Y/N)  OPTION _ (F/L)
  *             CONCATK _ (Y/N)  PARENTG _ (Y/N)  LOCKED _ (Y/N)

EX DL/I: VARLTH _ (Y/N)   OFFSET ___

WHERE/BOOLEAN SSA:                            BOOL
U IMSKEY OP   KEY                           OP
_ TRGEMKEY &caret.= 'SMITH'_______________________________________________   AND
_ TRGEMKEY &caret.= 'JONES'_____________________________________________   ___
_ ________ __ _____________________________________________________   ___
_ ________ __ _____________________________________________________   ___
_ ________ __ _____________________________________________________   ___
_ ________ __ _____________________________________________________   ___
_ ________ __ _____________________________________________________   ___
_ ________ __ _____________________________________________________   ___
_ ________ __ _____________________________________________________   ___
```

Notice that when SAM was updated, the entire list of DSC names was realphabetized (putting SAM before SAMPLE), as shown in the following example.

Type an **S** to the left of the new DSC and press Enter. Then press PF3 to save this information.

```
SELECT —— UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000006 COL 01
COMMAND ===>                          SCROLL ===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL

****** DLIDSC  USECNT CMND  IMSKEY  OP    KEY           MORE
****** ******** ****** **** ******** ** ****************************** *
000001 **DFLT**     *--- TRGEMKEY =  XFER-EMPL-ID
000002 GREATER  000008 *--- TRGEMKEY >  TPI-ID
000003 GREATERP IMPORT *--- TRGEMKEY >  TPI_ID
000004 GTEQ   000003 *--- TRGEMKEY >= TPI-ID
S 0005 SAM    TRCC2D *--- TRGEMKEY &caret.= 'SMITH'            +
000006 SAMPLE  000001 *--- TRGEMKEY =  XFER-EMPL-ID         +
****** ******** ****** **** ******** ** ****************************** *
```

CA Telon displays the DL/I Detail Data Access screen. Notice how the message **DLIDSC INFORMATION PROCESSED** is displayed in the upper right-hand corner. Notice also that the DSC name was changed to the selected DSC name "SAM".

From the DL/I Detail Data Access screen, type a non-blank character in the OPTIONS field after the PREVIEW literal and press Enter to preview the segment (in this case, TRGEMPL) as shown in the following example.

```
TRCC2D.SD AUTOEXEC OUTREAD TRGEMPL * ************ DLIDSC INFORMATION PROCESSED
COMMAND ==> _________________________________________________________________
OPTIONS ==> PREVIEW X

GENERAL: FUNC  ________ SSALIST _ (Y/N)  CONCATK _ (Y/N)
  *   IGNORE ___________________________________________________________
  *   IOAREA @___________________________________________________________
  *   PATH  ________ PARENTG ________ LOCKED ________ CURRENT ________
** DSCREF SEGMENT CMND  IMSKEY  OP    KEY
01 SAM_____ TRGEMPL_ @*--- @TRGEMKEY &caret.= @'SMITH'______________________ _
```

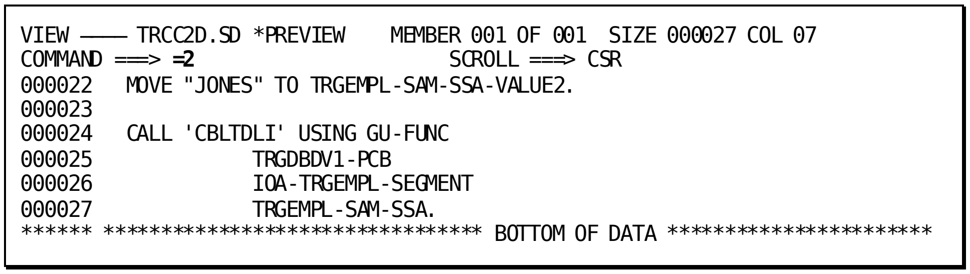
CA Telon displays the DL/I Preview screen. Notice that the Boolean operator **&.**, for AND, is shown, but the Boolean operator **+**, for OR, is not shown.

```
VIEW ---- TRCC2D.SD *PREVIEW    MEMBER 001 OF 001  SIZE 000027 COL 07
COMMAND ===>                              SCROLL ===> CSR
****** ********************************* TOP OF DATA **************************
000001 *  DEFAULT GENERATED USER I/O
000002 *  DEFAULT CALL: AUTOEXEC OUTREAD TRGEMPL *
000003
000004 *  ** SSAS IN STORAGE
000005
000006   05 TRGEMPL-SAM-SSA.
000007    10 TRGEMPL-SAM-SSA-SEGMENT    PIC X(8) VALUE 'TRGEMPL'.
000008    10 TRGEMPL-SAM-SSA-CMDCODE    PIC X(4) VALUE '*---'.
000009    10 TRGEMPL-SAM-SSA-LPAREN     PIC X(1) VALUE '('.
000010    10 TRGEMPL-SAM-SSA-IMSKEY1    PIC X(8) VALUE 'TRGEMKEY'.
000011    10 TRGEMPL-SAM-SSA-OPCODE1    PIC X(2) VALUE '&caret.='.
000012    10 TRGEMPL-SAM-SSA-VALUE1     PIC X(006).
000013    10 TRGEMPL-SAM-SSA-BOOLOP1    PIC X(1) VALUE '&'.
000014    10 TRGEMPL-SAM-SSA-IMSKEY2    PIC X(8) VALUE 'TRGEMKEY'.
000015    10 TRGEMPL-SAM-SSA-OPCODE2    PIC X(2) VALUE '&caret.='.
000016    10 TRGEMPL-SAM-SSA-VALUE2     PIC X(006).
000017    10 FILLER             PIC X(2) VALUE ')'.
000018
000019 *  ** ASSIGNMENT STATEMENT FOR KEY
000020
000021   MOVE 'SMITH' TO TRGEMPL-SAM-SSA-VALUE1.
```

Press PF8 to page down.

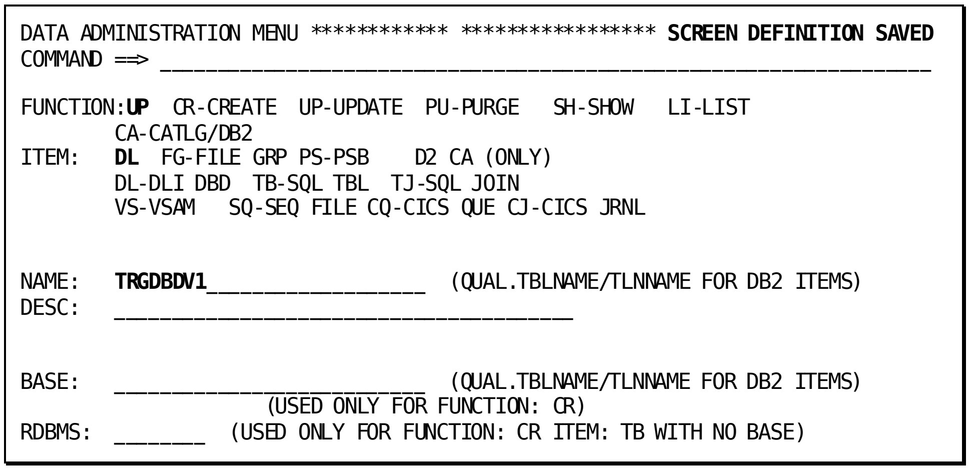
The following example shows the rest of the code generated for the new DSC SAM.

Type **=2** in the COMMAND line to return to the Data Administration menu.



```
VIEW ---- TRCC2D.SD *PREVIEW    MEMBER 001 OF 001  SIZE 000027 COL 07
COMMAND ===> =2                       SCROLL ===> CSR
000022   MOVE "JONES" TO TRGEMPL-SAM-SSA-VALUE2.
000023
000024   CALL 'CBLTDLI' USING GU-FUNC
000025               TRGDBDV1-PCB
000026               IOA-TRGEMPL-SEGMENT
000027               TRGEMPL-SAM-SSA.
****** ******************************** BOTTOM OF DATA **********************
```

You can now update the DL/I DBD. From the Data Administration menu, type **UP** in the FUNCTION field, **DL** in the ITEM field, the DL/I DBD name in the NAME field (in this case, **TRGDBDV1**), and press Enter.



```
DATA ADMINISTRATION MENU ************ ***************** SCREEN DEFINITION SAVED
COMMAND ==> _________________________________________________________________

FUNCTION:UP  CR-CREATE  UP-UPDATE  PU-PURGE   SH-SHOW   LI-LIST
        CA-CATLG/DB2
ITEM:   DL  FG-FILE GRP PS-PSB    D2 CA (ONLY)
        DL-DLI DBD  TB-SQL TBL  TJ-SQL JOIN
        VS-VSAM   SQ-SEQ FILE CQ-CICS QUE CJ-CICS JRNL


NAME:   TRGDBDV1__________________  (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:   _______________________________________


BASE:   __________________________  (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                     (USED ONLY FOR FUNCTION: CR)
RDBMS:  ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update DBD screen.

From the Update DBD screen, type a **U** to the left of the SEGMENT you want to update (in this case, TRGEMPL) as shown in the following example.

```
TRGDBDV1 UPDATE DBD ****************
********************************************************************************
COMMAND ==> ______________________________________________________ PAGE 01

ACCESS HIDAM,VSAM       RMNAME

SEQ  TYPE      NAME   PARENT/DEVICE     MAX LTH SEGMENT KEY LENGTH  START
 001 DATASET TRGDBDV1   3390___                  _____  ________  _____
      _____
 U 2 SEGM___ TRGEMPL_   0_______        600     TRGEMKEY     6           1
 003 LCHILD_  TRGINDX_          TRGDBDVX        600     TRGEMKEY     6   1
 004 SEGM___ TRGTASK_   TRGEMPL_        600     TRGTAKEY      8  1
 005 SEGM___ TRGTIME_   TRGTASK_        600     TRGTIKEY       2    1
```

CA Telon displays the Update DBD Segment Defaults screen. Type a **U** in the DLIDSC field and press Enter.

```
UPDATE DBD: TRGDBDV1 SEGM: TRGEMPL
**************************************************************************

COMMAND ==>
__________________________________________________________________________
OPTIONS ==>   SEARCH FIELDS +   PCB PARMS +     DLIDSC U

GENERAL: LABEL   ________
  *       COPY   TRGEMPL_       COPYLV1 _   COPYLBL ______________________________

**    DLIDSC    SEGMENT  CMND    IMSKEY          OP      KEY
01   **DFLT**   TRGEMPL  *---    TRGEMKEY        =_  XFER-EMPL-ID_______________
_
```

CA Telon displays the Update SSAs screen. From the Update SSAs screen, press PF3 twice to return to the Data Administration menu.

```
EDIT ---- UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000006 COL 01
COMMAND ==>                          SCROLL ===> CSR
DBD: TRGDBDV1  SEGMENT: TRGEMPL

****** DLIDSC  USECNT CMND  IMSKEY  OP    KEY           MORE
****** ******** ****** **** ******** ** ****************************** *
000001 **DFLT**     *--- TRGEMKEY =  XFER-EMPL-ID
000002 GREATER  000008 *--- TRGEMKEY >  TPI-ID
000003 GREATERP IMPORT *--- TRGEMKEY >  TPI_ID
000004 GTEQ   000003 *--- TRGEMKEY >= TPI-ID
000005 SAM    TRCC2D *--- TRGEMKEY &caret.= 'SMITH'            +
000006 SAMPLE  000001 *--- TRGEMKEY =  XFER-EMPL-ID         +
****** ******** ****** **** ******** ** ****************************** *
```

Conclusion

The point of this Updating DL/I DSCs (Boolean Example) sample session was to show that any new DLIDSCs created at the program level are also saved in data administration. Therefore, the new DLIDSCs are shareable between the program level and data administration.

Updating Secondary Indexes

From the TDF Main menu, type **2** in the FUNCTION field and press Enter. CA Telon displays the Data Administration menu.

```
TELON DESIGN FACILITY MAIN MENU ***** ****************************************
COMMAND ==> __________________________________________________________________

FUNCTION: 2_

      1 -- USER PROFILE MAINTENANCE
      2 -- DATA ADMINISTRATION
      3 -- PANEL SPECIFICATION
      4 -- ONLINE PROGRAM DEFINITION
      5 -- BATCH PROGRAM DEFINITION
      6 -- PROTOTYPING FACILITY
      U -- UTILITIES
      X -- EXIT
```

To update a secondary index DBD that was previously imported or created, perform the following procedure.

From the Data Administration menu, type **UP** in the FUNCTION field, **DL** in the ITEM field, and the secondary index name in the NAME field (in this case, **TRGDBDV1**) and press Enter.

```
DATA ADMINISTRATION MENU ************ ****************************************
 COMMAND ==> _________________________________________________________________

 FUNCTION:UP   CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:     DL  FG-FILE GRP      PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD      TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ   FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:    TRGDBDV1________________________       (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
 DESC:    ________________________________________


 BASE:    ________________________________       (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                         (USED ONLY FOR FUNCTION: CR)
 RDBMS:           ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

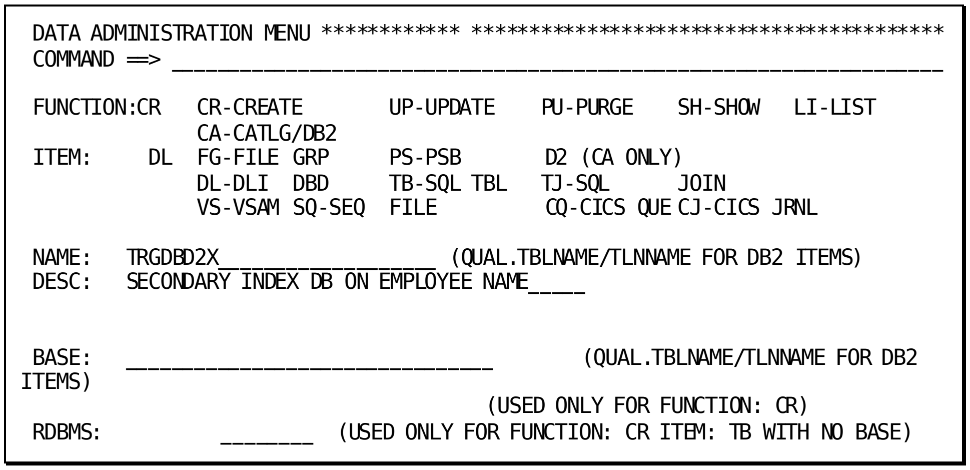
CA Telon displays the Update DBD screen. From the Update DBD screen, you can update DBD information for segments associated with this DBD.

```
 TRGDBDV1 UPDATE DBD **************** ****************************************
 COMMAND ==> _____________________________________________________ PAGE 01

 ACCESS HIDAM,VSAM       RMNAME

 SEQ  TYPE     NAME   PARENT/DEVICE     MAX LTH SEGMENT KEY LENGTH  START
 001 DATASET TRGDBDV1  3390____                  _____  ________  _____
      _____
 002 SEGM___ TRGEMPL_  0_______         600     TRGEMKEY     6          1
 003 LCHILD_ TRGINDX_  TRGDBDVX         _____   ________  _____        _____
 004 LCHILD_ TRG2NDX_  TRGDBD2X         _____   X2NAME__    25   _____
 005 SEGM___ TRGTASK_  TRGEMPL_         600     TRGTAKEY     8          1
 006 SEGM___ TRGTIME_  TRGTASK_         600     TRGTIKEY     2          1
```

Press PF3 to return to the Data Administration menu.

To create a secondary index DBD definition, perform the following procedure.

From the Data Administration menu, type **CR** in the FUNCTION field, **DL** in the ITEM field, the secondary index name in the NAME field (in this case, **TRGDBD2X**), a description in the DESC field, and press Enter.

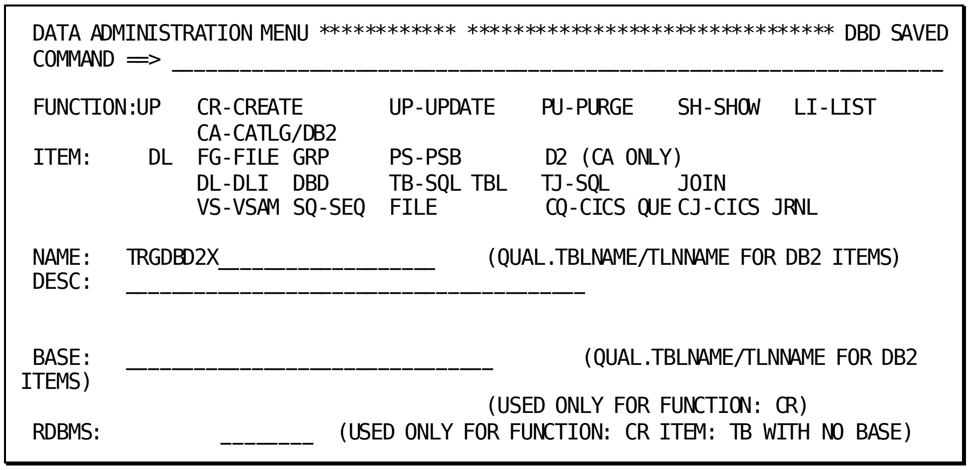


```
 DATA ADMINISTRATION MENU ************ ****************************************
 COMMAND ==> __________________________________________________________________

 FUNCTION:CR   CR-CREATE       UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2
 ITEM:     DL  FG-FILE GRP     PS-PSB       D2 (CA ONLY)
               DL-DLI  DBD     TB-SQL TBL   TJ-SQL      JOIN
               VS-VSAM SQ-SEQ  FILE         CQ-CICS QUE CJ-CICS JRNL

 NAME:   TRGDBD2X__________________ (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
 DESC:   SECONDARY INDEX DB ON EMPLOYEE NAME____


 BASE:   ______________________________       (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                        (USED ONLY FOR FUNCTION: CR)
 RDBMS:          ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Create DBD screen. From the Create DBD screen, type the SEGMENT information in the TYPE, NAME, PARENT/DEVICE, MAX LTH, SEGMENT KEY, LENGTH, and START fields as shown in the following example. Press PF3 to return to the Data Administration menu.

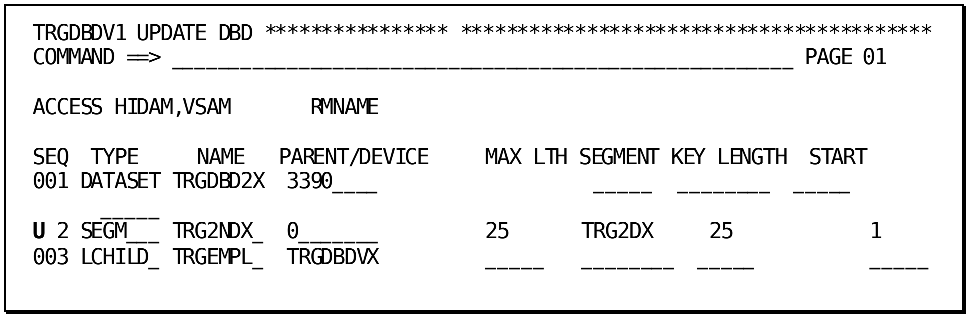
```
TRGDBD2X CREATE DBD **************** ****************************************
COMMAND ==> _____________________________________________________ PAGE 01

ACCESS                                  RMNAME

SEQ    TYPE    NAME       PARENT/DEVICE        MAX LTH SEGMENT KEY      LENGTH
START
001    DATASET TRGDBD2X  3390____       _____   ________  _____         _____
       _____
002    SEGM___  TRG2NDX_   0_______     25___   TRG2DX  25___   1____   _____
003    LCHILD_  TRGEMPL_    TRGDBDV1    _____    _____  ________  _____
       _____
004    _______ ________    ________     _____    _____           _____   _____
       _____
005    _______ ________    ________     _____    _____           _____   _____
       _____
006    _______ ________    ________     _____    _____           _____   _____
       _____
007    _______ ________    ________     _____    _____           _____   _____
       _____
008    _______ ________    ________     _____    _____           _____   _____
       _____
009    _______ ________    ________     _____    _____           _____   _____
       _____
010    _______ ________    ________     _____    _____           _____   _____
       _____
011    _______ ________    ________     _____    _____           _____   _____
       _____
012    _______ ________    ________     _____    _____           _____   _____
       _____
013    _______ ________    ________     _____    _____           _____   _____
       _____
014    _______ ________    ________     _____    _____           _____   _____
       _____
015    _______ ________    ________     _____    _____           _____   _____
       _____
016    _______ ________    ________     _____    _____           _____   _____
       _____
017    _______ ________    ________     _____    _____           _____   _____
       _____
018    _______ ________    ________     _____    _____           _____   _____
       _____
```

From the Data Administration Menu, type **UP** in the FUNCTION field, **DL** in the ITEM field, the secondary index name in the NAME field (in this case, **TRGDBD2X**), and press Enter.



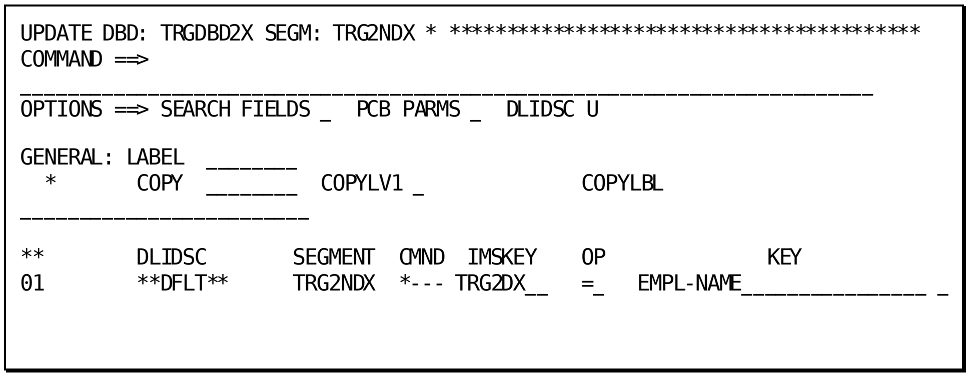
```
 DATA ADMINISTRATION MENU ************ ******************************** DBD SAVED
 COMMAND ==> ____________________________________________________________________

 FUNCTION:UP    CR-CREATE        UP-UPDATE     PU-PURGE    SH-SHOW   LI-LIST
                CA-CATLG/DB2
 ITEM:     DL   FG-FILE GRP      PS-PSB         D2 (CA ONLY)
                DL-DLI  DBD      TB-SQL TBL    TJ-SQL      JOIN
                VS-VSAM SQ-SEQ   FILE           CQ-CICS QUE CJ-CICS JRNL

 NAME:   TRGDBD2X___________________     (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
 DESC:   ________________________________________


 BASE:   ________________________________        (QUAL.TBLNAME/TLNNAME FOR DB2
ITEMS)
                                         (USED ONLY FOR FUNCTION: CR)
 RDBMS:          ________   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update DBD screen. From the Update DBD screen, type **U** next to the segment you want to update and press Enter.



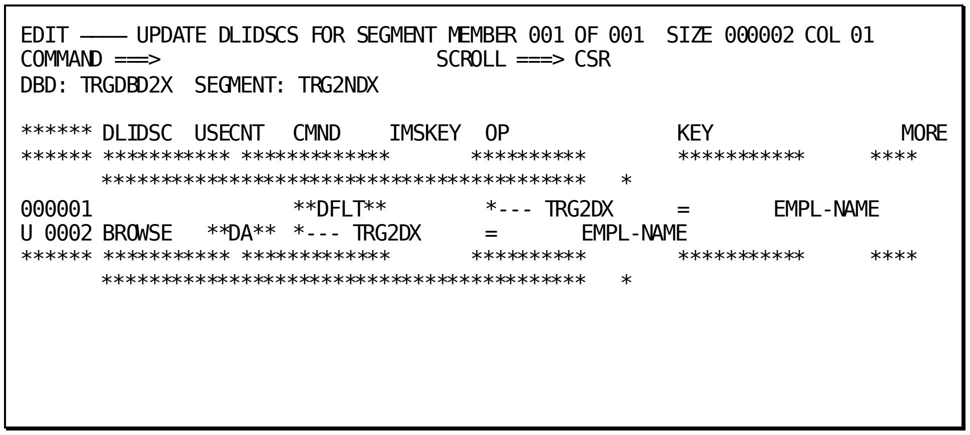
```
 TRGDBDV1 UPDATE DBD **************** *****************************************
 COMMAND ==> ______________________________________________________ PAGE 01

 ACCESS HIDAM,VSAM       RMNAME

 SEQ  TYPE     NAME   PARENT/DEVICE     MAX LTH SEGMENT KEY LENGTH  START
 001 DATASET TRGDBD2X  3390____                   _____  ________  _____
        _____
 U 2 SEGM___  TRG2NDX_  0_______          25      TRG2DX     25           1
 003 LCHILD_  TRGEMPL_  TRGDBDVX          _____   ________  _____         _____
```

CA Telon displays the Update Database Segment screen. From the Update Database Segment screen, type a **U** in the DLIDSC field and press Enter.



```
UPDATE DBD: TRGDBD2X SEGM: TRG2NDX * *****************************************
COMMAND ==>
___________________________________________________________________________
OPTIONS ==> SEARCH FIELDS _   PCB PARMS _   DLIDSC U

GENERAL: LABEL  ________
  *       COPY  ________  COPYLV1 _            COPYLBL
_________________________

**        DLIDSC       SEGMENT  CMND  IMSKEY    OP              KEY
01        **DFLT**     TRG2NDX  *--- TRG2DX__   =_   EMPL-NAME________________ _
```

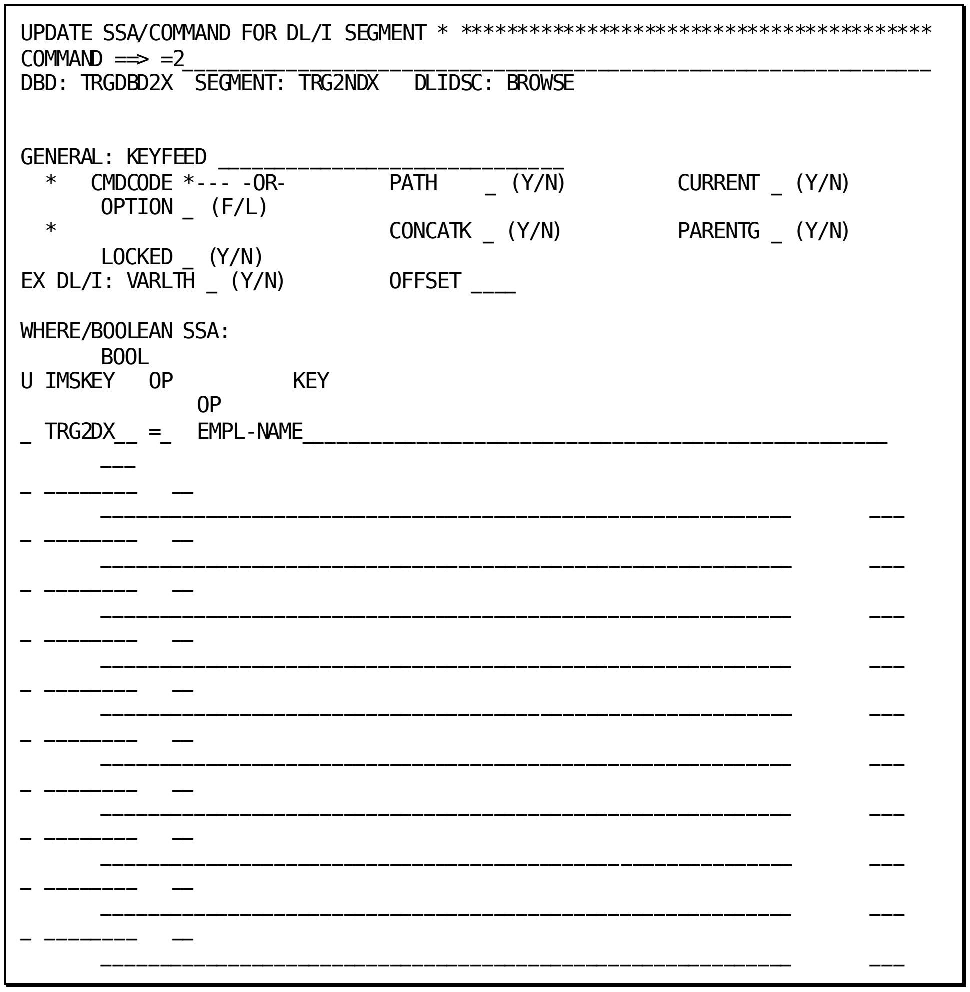
CA Telon displays the Update SSAs screen. From the Update SSAs screen type **U** next to the SSA you want to update and press Enter.



```
EDIT ———— UPDATE DLIDSCS FOR SEGMENT MEMBER 001 OF 001  SIZE 000002 COL 01
COMMAND ===>                            SCROLL ===> CSR
DBD: TRGDBD2X  SEGMENT: TRG2NDX

****** DLIDSC  USECNT   CMND    IMSKEY  OP               KEY               MORE
****** *********** *************       **********        ***********      ****
       ******************************************   *
000001                  **DFLT**         *--- TRG2DX      =       EMPL-NAME
U 0002 BROWSE   **DA**  *--- TRG2DX      =       EMPL-NAME
****** *********** *************       **********        ***********      ****
       ******************************************   *
```

CA Telon displays the Create/Update SSAs screen.

Type **=2** in the COMMAND line to return to the Data Administration Menu.



```
UPDATE SSA/COMMAND FOR DL/I SEGMENT * ****************************************
COMMAND ==> =2______________________________________________________________
DBD: TRGDBD2X  SEGMENT: TRG2NDX   DLIDSC: BROWSE


GENERAL: KEYFEED _____________________________
  *   CMDCODE *--- -OR-         PATH    _ (Y/N)         CURRENT _ (Y/N)
      OPTION _  (F/L)
  *                             CONCATK _ (Y/N)         PARENTG _ (Y/N)
      LOCKED _ (Y/N)
EX DL/I: VARLTH _ (Y/N)         OFFSET ___

WHERE/BOOLEAN SSA:
      BOOL
U IMSKEY   OP          KEY
              OP
_ TRG2DX__ =_  EMPL-NAME_________________________________________________
      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
_ ________   __
      ___________________________________________________________      ___
```

Chapter 5: VSAM Tasks

This chapter takes you through common CA Telon administrative tasks performed using VSAM.

The first subject VSAM Access describes important concepts pertaining to VSAM access. The later subjects give sample sessions for doing common VSAM tasks, with appropriate explanations.

VSAM Access

Considerations unique to VSAM file processing are discussed in this topic. You can implement these concepts using the Create/Update Data Group screen and the Update VSAM/SEQ Detail Data Access screen. These screens are described in the *Design Facility Reference Guide*.

File Group

The initial specification of VSAM file accesses can be assisted by creating a file group in CA Telon TDF option 2. This file group contains a list of all VSAM files, databases and DB2 tables that an application is likely to use. Instead of remembering the file names, you can simply enter auto exec and user exec statements under the line for the proper VSAM file.

You can bring a list of files, databases and tables into your program definition in two ways:

- When creating the initial program definition, identify a BASE that already has a data group
- From the Create/Update Data Group screen, enter the DGADD command and copy in file, database, or table information from a PSB, a file group, another program definition, a DBD definition, a VSAM file definition or a DB2 definition

Refer to the description of the Create/Update Data Group screen in the *Design Facility* Reference *Guide* for further explanation of the DGADD command.

Access Specification

You specify generated data access on the Create/Update Data Group screen for each VSAM file. Each file is defined on a line of the Create/Update Data Group screen and identified by the characters VSAM=> in the first field on the line. A VSAM file line exists due to:

- The use of a BASE from another program definition
- A DGADD command
- The repeat or copy of another VSAM line in this program definition

  The VSAM file line contains one additional field besides the VSAM=> indication: the VSAM file name as defined to CICS (for online) or OS/390 and z/OS (for batch). Note that the titles at the top of the screen refer to the auto exec and user exec lines, not to the VSAM file line. The VSAM file named on this line is used for all auto exec and user exec statements that are defined under it.

  Information about the record follows the VSAM=> line. It is identified by the characters REC=> in the first field on the line and created at the same time as the VSAM file line. The information about the record, such as COPY book name, can be modified from the Update Record screen. You access this screen by entering a **U** on the REC=> line to access the Update Record screen.

  You enter each auto exec and user exec statement on a line under the definition for the VSAM file to be accessed. The general parameters for each access definition are described under Generated Data Access in the chapter "Data Access Concepts" in this guide. Those parameters cause the generation of appropriate EXEC CICS statements (for online) or VSAM I/O statements (for batch).

  For online, if you require more control of the file access, you can specify lower level parameters that define characteristics for the EXEC CICS statement generated. For example, you can use the OPTLIST parameter to extend the exec CICS statement with CICS options. You specify these parameters on the Update VSAM/SEQ Detail Data Access screen. You access this screen from the Create/Update Data Group screen.

Alternate Indices

CA Telon treats alternate indices as separate file names in the TDF, the same as in native CICS or OS/390 and z/OS. Thus, to access a file through an alternate index, that alternate index must have been set up under a separate name to CICS or OS/390 and z/OS. When accessing a file through its alternate index, set up a VSAM file entry in the file group or on the Create/Update Data Group screen, using the alternate index name given to CICS or OS/390 and z/OS. Auto execs and user execs are then defined under that VSAM file line for the alternate index, as if it were a separate file.

Generic Keys (Online Only)

With CICS, you have a capability for sequential processing not available in all access methods. This generic key capability allows you to define a generic key length for the auto exec BROWSE, the user exec READNEXT, and the user exec READ with a FUNC of STARTBR or READNEXT.

When you do not specify a key length, records continue to be read sequentially with each access until an ENDFILE return indication is set. You must check each record returned in custom code to ensure that it meets the key criteria for records to process.

With a generic key length, you can identify that part of the key which must match the value of a variable, for a record to be returned. If the next record read by CICS does not match the variable for the length of the generic key, CICS returns a NOT FOUND condition.

These parameters are specified on the Update VSAM/SEQ Detail Data Access screen.

Variable Length Records (Online Only)

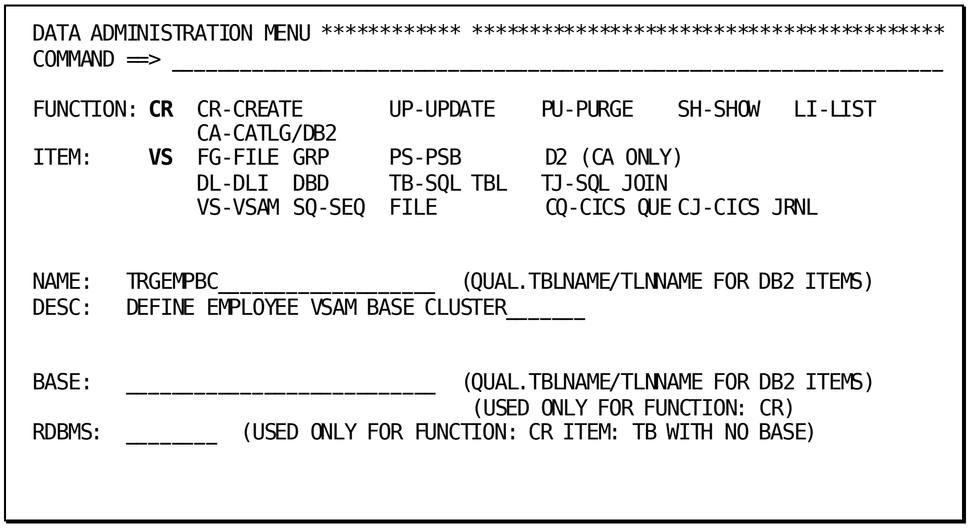
CA Telon supports CICS VSAM variable length records. To use variable length records, do the following:

- Make sure that any COBOL or PL/I record definitions are large enough to handle the largest record to be accessed
- Identify the maximum length of the record to be read in the RECLTH parameter on the Update VSAM/SEQ Detail Data Access screen by either entering the maximum length in the RECLTH parameter or specifying a variable name that will contain the maximum length during execution
- Prior to any auto exec or user exec writes, set up the length of the record in the I/O area for the record

Except for these considerations, CA Telon handles variable length records the same as regular records.

Creating VSAM File Definitions

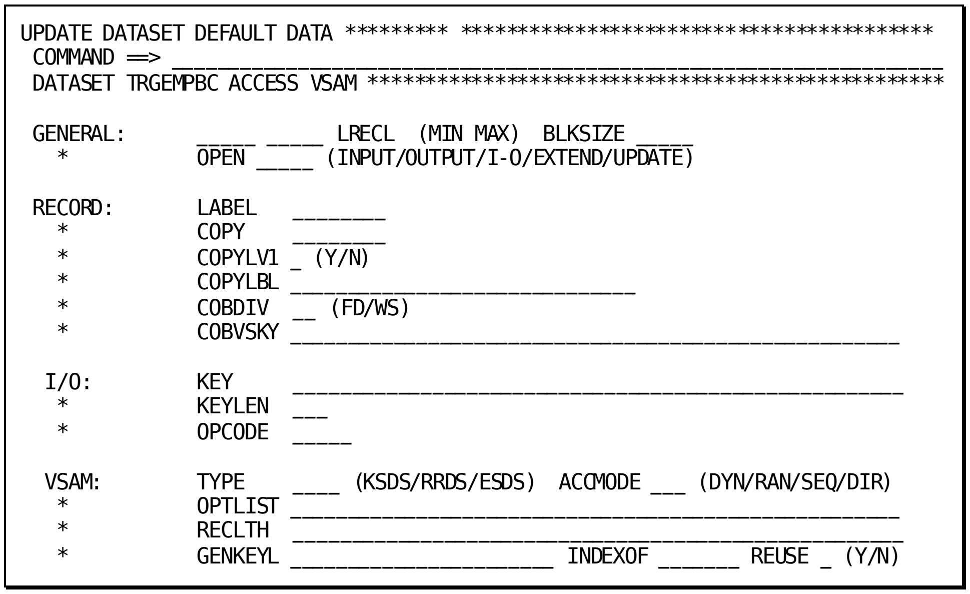
You can create VSAM file definitions by first accessing the TDF Main menu by using the procedures at a place on your site. From this menu, type **2** in the FUNCTION field and press Enter. to display the Data Administration menu.



```
DATA ADMINISTRATION MENU ************ ******************************************
COMMAND ==> __________________________________________________________________

FUNCTION: CR  CR-CREATE       UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
              CA-CATLG/DB2
ITEM:     VS  FG-FILE GRP     PS-PSB       D2 (CA ONLY)
              DL-DLI  DBD     TB-SQL TBL   TJ-SQL JOIN
              VS-VSAM SQ-SEQ  FILE         CQ-CICS QUE CJ-CICS JRNL


NAME:    TRGEMPBC__________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    DEFINE EMPLOYEE VSAM BASE CLUSTER______


BASE:    ___________________________  (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)
RDBMS:   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

From the Data Administration menu, type **CR** in the FUNCTION field, **VS** in the ITEM field, and VSAM Table Name in the NAME field (in this case, **TRGEMPBC**), a description in the DESC field, and press Enter.

CA Telon displays the Update Dataset Default Data screen.

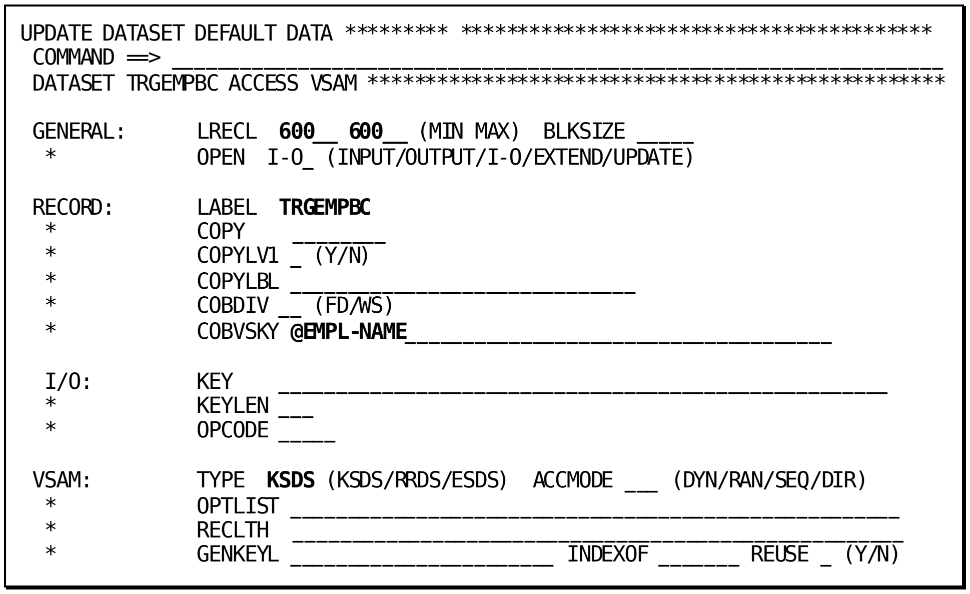


```
UPDATE DATASET DEFAULT DATA ********* ****************************************
 COMMAND ==> __________________________________________________________________
 DATASET TRGEMPBC ACCESS VSAM **************************************************

 GENERAL:      _____ _____ LRECL  (MIN MAX)  BLKSIZE _____
   *           OPEN ____ (INPUT/OUTPUT/I-O/EXTEND/UPDATE)

 RECORD:       LABEL    ________
   *           COPY     ________
   *           COPYLV1 _ (Y/N)
   *           COPYLBL ______________________________
   *           COBDIV  __ (FD/WS)
   *           COBVSKY _____________________________________________________

  I/O:         KEY      _____________________________________________________
   *           KEYLEN   ___
   *           OPCODE   _____

  VSAM:        TYPE     ____ (KSDS/RRDS/ESDS)  ACCMODE ___ (DYN/RAN/SEQ/DIR)
   *           OPTLIST _____________________________________________________
   *           RECLTH   _____________________________________________________
   *           GENKEYL _______________________ INDEXOF _______ REUSE _ (Y/N)
```

From the Update Dataset Default Data screen:

1. Type the length of record (minimum and maximum) in the LRECL field (in this case, **600** and **600**).
2. Specify how you want CA Telon to automatically open the data set at program initialization by using the OPEN field (in this case, typing **I-O** for input/output).
3. Specify the data name of the key for the file in the COBVSKY field (in this case, typing *@EMPL-ID). The @ sign is* the symbol for inheritance. Refer to the Programming Concepts Guide for details about the concept of data inheritance.
4. Type **KSDS** in the TYPE field (for VSAM key-sequenced data set).
5. Press Enter. The Update Dataset Default Data screen is redisplayed.

Type a label in the LABEL field (in this case, **TRGEMPBC**). When you specify a label, CA Telon replaces the name of the data set in the names that CA Telon generates.

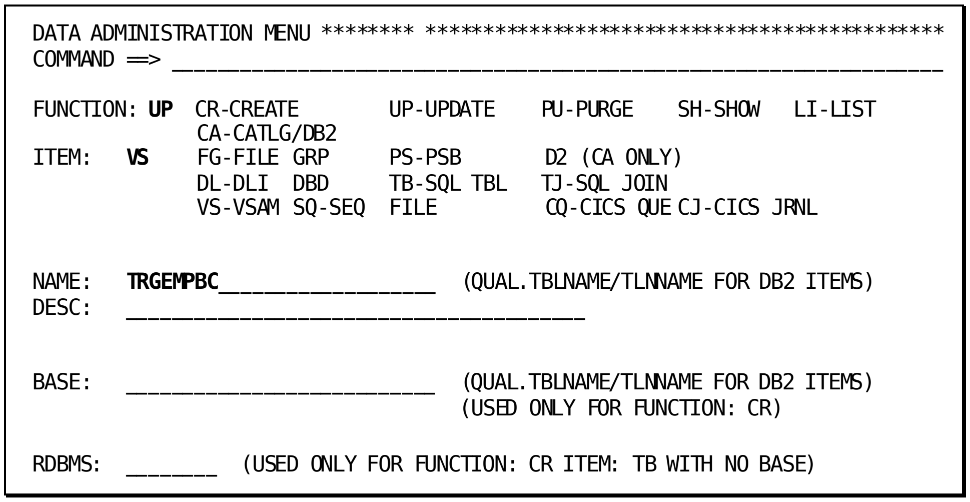


```
UPDATE DATASET DEFAULT DATA ********* ****************************************
 COMMAND ==> ____________________________________________________________
 DATASET TRGEMPBC ACCESS VSAM ***********************************************

 GENERAL:       LRECL  600__ 600__ (MIN MAX)  BLKSIZE _____
  *             OPEN  I-O_ (INPUT/OUTPUT/I-O/EXTEND/UPDATE)

 RECORD:        LABEL  TRGEMPBC
  *             COPY    ________
  *             COPYLV1 _ (Y/N)
  *             COPYLBL ______________________________
  *             COBDIV __ (FD/WS)
  *             COBVSKY @EMPL-NAME_____________________________________

  I/O:          KEY  ____________________________________________________
  *             KEYLEN ___
  *             OPCODE _____

 VSAM:          TYPE  KSDS (KSDS/RRDS/ESDS)  ACCMODE ___  (DYN/RAN/SEQ/DIR)
  *             OPTLIST _____________________________________________________
  *             RECLTH  _____________________________________________________
  *             GENKEYL _______________________ INDEXOF _______ REUSE _ (Y/N)
```

Press PF3 to update all information typed in. CA Telon returns you to the Data Administration menu with the message **DATASET DEFAULT INFORMATION UPDATED** in the upper right-hand corner.

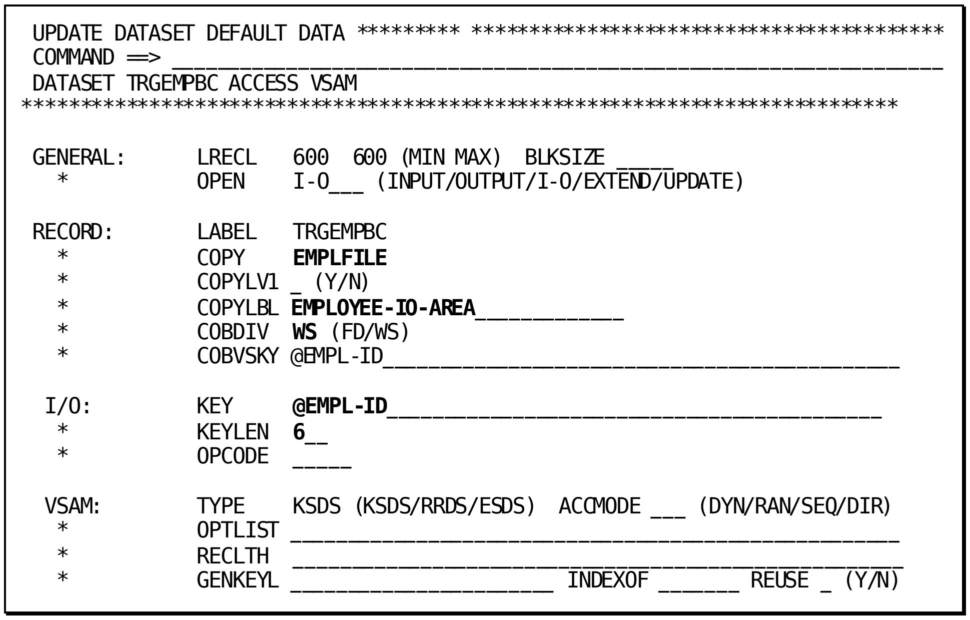
Updating VSAM File Definitions

You can update VSAM file definitions by accessing the Data Administration menu from the TDF Main menu. Type **UP** in the FUNCTION field, **VS** in the ITEM field, and the VSAM table name in the NAME field (in this case, **TRGEMPBC**) and press Enter.



```
DATA ADMINISTRATION MENU ******** ********************************************
COMMAND ==> __________________________________________________________________

FUNCTION: UP  CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
              CA-CATLG/DB2
ITEM:    VS   FG-FILE GRP      PS-PSB       D2 (CA ONLY)
              DL-DLI  DBD      TB-SQL TBL   TJ-SQL JOIN
              VS-VSAM SQ-SEQ   FILE         CQ-CICS QUE CJ-CICS JRNL


NAME:    TRGEMPBC__________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    ________________________________________


BASE:    __________________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                      (USED ONLY FOR FUNCTION: CR)

RDBMS:   ________   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update Dataset Default Data screen. Make the updates desired.



```
UPDATE DATASET DEFAULT DATA ********* ****************************************
COMMAND ==> __________________________________________________________________
DATASET TRGEMPBC ACCESS VSAM
****************************************************************************

GENERAL:      LRECL   600  600 (MIN MAX)  BLKSIZE _____
  *           OPEN    I-O___ (INPUT/OUTPUT/I-O/EXTEND/UPDATE)

RECORD:       LABEL   TRGEMPBC
  *           COPY    EMPLFILE
  *           COPYLV1 _ (Y/N)
  *           COPYLBL EMPLOYEE-IO-AREA____________
  *           COBDIV  WS (FD/WS)
  *           COBVSKY @EMPL-ID_____________________________________________

 I/O:         KEY     @EMPL-ID____________________________________________
  *           KEYLEN  6__
  *           OPCODE  _____

 VSAM:        TYPE    KSDS (KSDS/RRDS/ESDS)  ACCMODE ___ (DYN/RAN/SEQ/DIR)
  *           OPTLIST _____________________________________________________
  *           RECLTH  _____________________________________________________
  *           GENKEYL ______________________ INDEXOF _______ REUSE _ (Y/N)
```

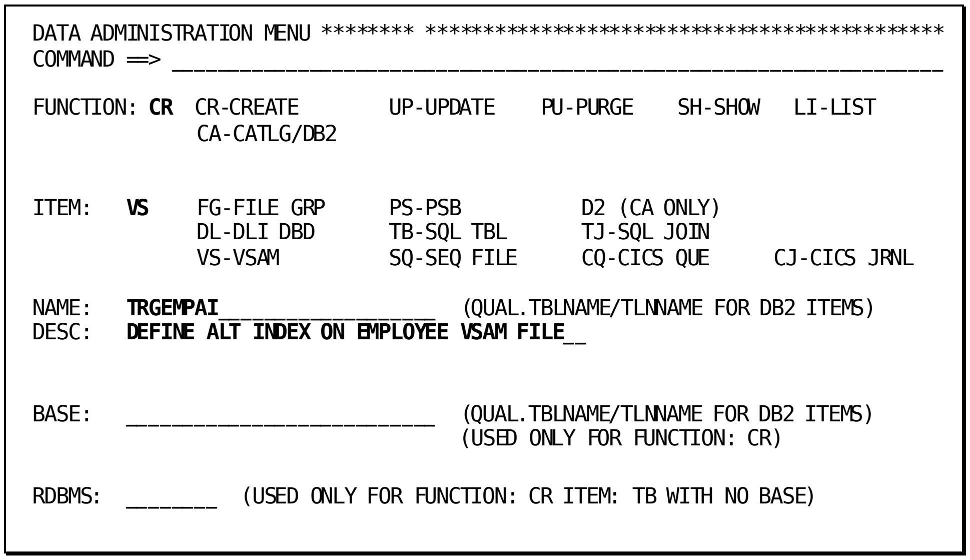
In this example, five common updates are shown:

- Type a Copy member name in the COPY fields (in this case, **EMPLFILE**). CA Telon uses the contents of this specified copy member name for the layout of the records.
- Type a parameter in the COPYLBL field (in this case, **EMPLOYEE-IO-AREA**).
- Type **WS** in the COBDIV field to specify that the record layout appears in the IO-AREA of the program.
- Specify a key in the KEY field (in this case, **@EMP-ID**) to specify the variable names containing the key to a record (in this case, COBOL).
- Specify the length of the RID (Record Identification) field in the KEYLEN field (in this case, **6**).

Press PF3 to update this information. After you press PF3 on the Update Dataset Default Data screen, you return to the Data Administration menu with the VSAM file definition updated. CA Telon confirms this with the message **DATASET DEFAULT INFORMATION UPDATED** in the upper right-hand corner of the screen.

Creating Alternate Indexes

You can create alternate indexes by accessing the Data Administration menu from the TDF Main menu. Type **CR** in the FUNCTION field, **VS** in the ITEM field, an alternate index name in the NAME field (in this case, **TRGEMPAI**), a description in the DESC field, and press Enter.

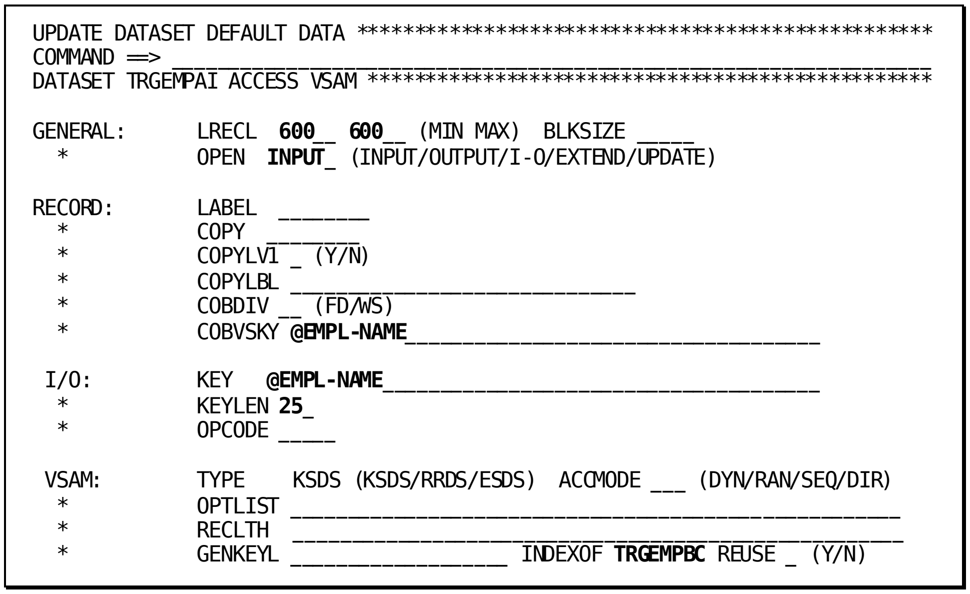


```
DATA ADMINISTRATION MENU ******** *********************************************
COMMAND ==> ___________________________________________________________________

FUNCTION: CR  CR-CREATE       UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
              CA-CATLG/DB2


ITEM:    VS   FG-FILE GRP     PS-PSB         D2 (CA ONLY)
              DL-DLI DBD      TB-SQL TBL     TJ-SQL JOIN
              VS-VSAM         SQ-SEQ FILE    CQ-CICS QUE     CJ-CICS JRNL

NAME:    TRGEMPAI__________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    DEFINE ALT INDEX ON EMPLOYEE VSAM FILE__


BASE:    __________________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                      (USED ONLY FOR FUNCTION: CR)

RDBMS:   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

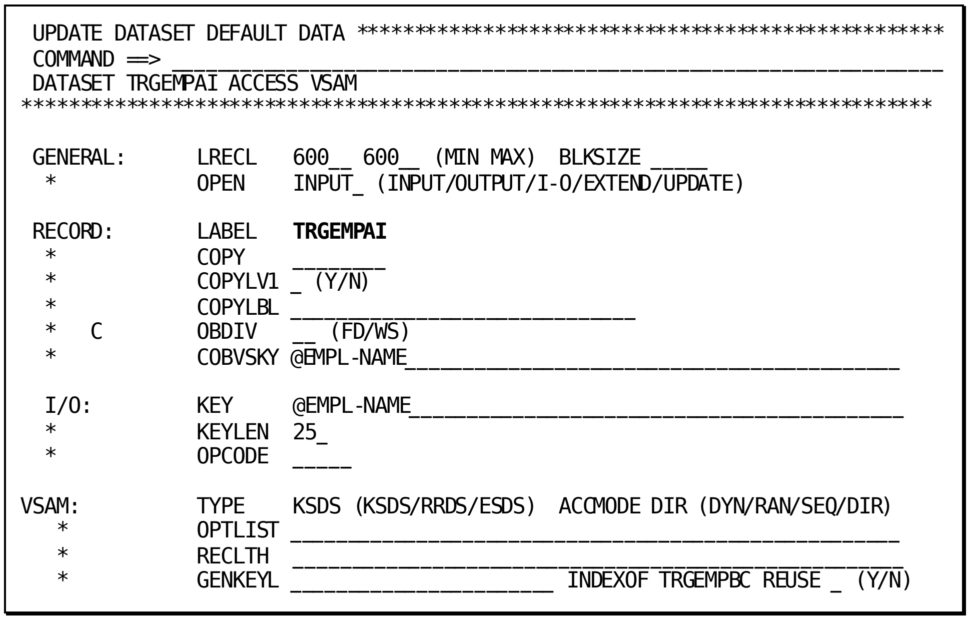
CA Telon displays the Update Dataset Default Data screen.

From the Update Dataset Default Data screen:

1. Type the length of record (minimum and maximum) in the LRECL field (in this case, **600** and **600**).
2. Specify how you want CA Telon to automatically open the data set at program initialization by using the OPEN field (in this case, by typing **INPUT** for Input).
3. Specify the data name of the key for the file in the COBVSKY field (in this case, by typing **@EMPL-NAME**). The *@ sign is the* symbol for inheritance. Refer to the Programming Concepts Guide for details about the concept of data inheritance.
4. Specify a key in the KEY field (in this case, **@EMPL-NAME**) to specify the variable names containing the key to a record (in this case, COBOL).
5. Specify the length of the RID (Record Identification) field in the KEYLEN field (in this case, **25**).
6. Type **KSDS** in the TYPE field (for VSAM key-sequenced data set).
7. Specify how you want to access the type of data set you are defining (in this case, by typing **DIR** for VSAM direct access).
8. The INDEXOF parameter specifies that this data set is an index of another VSAM file definition. In this case, type **TRGEMPBC** in the INDEXOF field to make TRGEMPAI an alternate index for the Employee VSAM base cluster (TRGEMPBC).
9. Press Enter. The Update Dataset Default Data screen redisplays.


```
UPDATE DATASET DEFAULT DATA ************************************************
COMMAND ==> ________________________________________________________________
DATASET TRGEMPAI ACCESS VSAM ***********************************************

GENERAL:      LRECL  600__ 600__ (MIN MAX)  BLKSIZE _____
  *           OPEN  INPUT_ (INPUT/OUTPUT/I-O/EXTEND/UPDATE)

RECORD:       LABEL  ________
  *           COPY  ________
  *           COPYLV1 _ (Y/N)
  *           COPYLBL ______________________________
  *           COBDIV __ (FD/WS)
  *           COBVSKY @EMPL-NAME____________________________________

 I/O:         KEY   @EMPL-NAME______________________________________
  *           KEYLEN 25_
  *           OPCODE _____

 VSAM:        TYPE    KSDS (KSDS/RRDS/ESDS)  ACCMODE ___ (DYN/RAN/SEQ/DIR)
  *           OPTLIST _____________________________________________________
  *           RECLTH  _____________________________________________________
  *           GENKEYL ___________________ INDEXOF TRGEMPBC REUSE _ (Y/N)
```

Type a label in the LABEL field (in this case **TRGEMPAI**). When you specify a label, CA Telon replaces the name of the data set in the names that CA Telon generates.


```
 UPDATE DATASET DEFAULT DATA ***************************************************
 COMMAND ==> ___________________________________________________________________
 DATASET TRGEMPAI ACCESS VSAM
********************************************************************************

 GENERAL:      LRECL   600__ 600__ (MIN MAX)  BLKSIZE _____
  *            OPEN    INPUT_ (INPUT/OUTPUT/I-O/EXTEND/UPDATE)

 RECORD:       LABEL   TRGEMPAI
  *            COPY    ________
  *            COPYLV1 _ (Y/N)
  *            COPYLBL ______________________________
  *   C        OBDIV   __ (FD/WS)
  *            COBVSKY @EMPL-NAME__________________________________________

  I/O:         KEY     @EMPL-NAME__________________________________________
  *            KEYLEN  25_
  *            OPCODE  _____

VSAM:          TYPE    KSDS (KSDS/RRDS/ESDS)  ACCMODE DIR (DYN/RAN/SEQ/DIR)
   *           OPTLIST _____________________________________________________
   *           RECLTH  _____________________________________________________
   *           GENKEYL _______________________ INDEXOF TRGEMPBC REUSE _ (Y/N)
```

Press PF3 to update all information typed in. CA Telon returns you to the Data Administration menu with the message **DATASET DEFAULT INFORMATION UPDATED** in the upper right-hand corner.

Chapter 6: CICS Tasks

This chapter takes you through common CA Telon administrative tasks performed for CICS tasks.

The first two subjects, queue access and journal access, describe important concepts pertaining to CICS access. The later subjects give sample sessions for doing common CICS tasks, with appropriate explanations.

Queue Access

The considerations unique to CICS queue processing are discussed in this section. You implement these concepts using the Create/Update Data Group screen and the Update CICS Queue Detail Data Access screen. These screens are described in the *Design Facility Reference Guide*.

CICS Queue

A CA Telon CICS queue defines CICS a transient data or temporary storage queue that can be referenced by CA Telon users during CICS program construction.

A CICS queue is a group of sequential records that can be accessed by multiple CICS transactions. There are two types of queues: transient data, and temporary storage. Transient data queues are physical sequential files used as either input or output to an application program. Temporary storage queues function as a scratch pad for an application program, and may reside in auxiliary storage or main memory.

Each CA Telon CICS queue definition is an independent object identified by the name of the queue. There are no sub-components.

Data administrators, designers, and programmers all use the CA Telon CICS queue definition to provide reusability of these objects within CA Telon file groups and programs.

File Group

The initial specification of CICS queue access can be assisted by creating a file group in TELON option 2. This file group contains a list of all VSAM files, databases, DB2 tables, and CICS queues and journals that an application is likely to use. Instead of remembering the file names, you can simply enter auto exec and user exec statements under the line for the proper CICS queue.

You can bring a list of files, databases, tables, queues, and journals into your program definition in two ways:

- When creating the initial program definition, identify a BASE that already has a data group
- From the Create/Update Data Group screen, enter the DGADD command and copy in file, database, or table information from a PSB, a file group, another program definition, a DBD definition, a VSAM File definition, a DB2 table definition, or a CICS queue or journal definition

See the description of the Create/Update Data Group screen in the *Design Facility* Reference *Guide* for further explanation of the DGADD command.

Access Specification

You specify data access on the Create/Update Data Group screen for each CICS queue. Each queue is defined on a line of the Create/Update Data Group screen and identified by the characters CQUE=> in the first field on the line. A CICS queue line exists due to:

- The use of a BASE from another program definition
- A DGADD command
- The repeat or copy of another VSAM line in this program definition.

The CICS queue line contains one additional field besides the CQUE=> indication: the queue name as defined to CICS. Note that the titles at the top of the screen refer to the auto exec and user exec lines, not to the CICS queue line. The CICS queue named on this line is used for all auto exec and user exec statements that are defined under it.

Information about the queue item follows the CQUE=> line. It is identified by the characters REC=> in the first field on the line and created at the same time as the CQUE file line. The information about the queue item, such as COPY book name, can be modified from the Update CICS Queue Record screen. You access this screen by entering a **U** on the REC=> line.

You enter each auto exec and user exec statement on a line under the definition for the CICS queue file to be accessed. The general parameters for each access definition are described under Generated Data Access in the "Data Access Concepts" chapter. Those parameters, which you enter on the Update CICS Queue Detail Data Access screen, cause the generation of appropriate EXEC CICS statements.

Segloop/Auto Exec Browse

Segloop/auto exec browse code generation is only supported for temporary storage queues. This function is not supported for transient data queues because CICS deletes each TD queue record as it is read, making paging back impossible. The Segloop logic generated for a CICS TS queues is a minor variation of that generated for any other data access type.

CA Telon generates CURRENT-SEGMENT-KEY and the PAGE-KEY-SAVE array as PIC 9(5) variables to hold a queue ITEM number. CA Telon assigns CURRENT-SEGMENT-KEY to the ITEM variable for each read of the queue. If paging is requested, the PAGE-KEY-SAVE array is used to hold ITEM numbers for each page. Any "start browse key" coded for a segloop should also be numeric.

CA Telon generates each READQ statement punched (one for "start browse," one for "read next") with the ITEM option, whether or not one is requested. You can still specify your own variable for the ITEM option, in the ITEMLBL parameter. Otherwise, CA Telon generates calls with ITEM (SYSWK-queuename-QUEUE-ITEM).

The "start browse" read for the first page of a paging segloop accesses a specific record. This record is the one with an item number identified in the variable named in the STBRKEY parameter on the Create/Update File SEGLOOP screen (If no variable is named, the record access is the first record on the queue, ITEM number one). Before each subsequent ("read next"), CA Telon increments the ITEM variable by one.

After a full page of queue records is read, one more read is done to get the first record for the next page. The ITEM number for that record is stored in CURRENT-SEGMENT-KEY, then reassigned to the ITEM variable before each subsequent "start browse" read.

ITEM numbers for the first ITEM on each page are stored in the PAGE-KEY-SAVE array, and used for paging backward. CA Telon determines the array size by the number of pages specified in the PAGESAV parameter on the Create/Update File SEGLOOP screen.

Note: You can only delete an entire queue. You cannot delete a single item. Also, you cannot insert a record into the center of a queue. Therefore, page positioning does not change as it would with any other kind of I/O.

Journal Access

Considerations unique to CICS journal processing are discussed in this section. You implement these concepts using the Create/Update Data Group screen and the Update CICS Journal Detail Data Access screen. These screens are described in the *Design Facility Reference Guide*.

CICS Journal

A CA Telon CICS journal defines CICS journals that can be referenced by CA Telon users during program construction.

A CICS journal is a chronological record of changes made to a recoverable resource. It allows CICS users to recover certain portions of a CICS system, instead of the entire system. After a system crash, a recovery program restores the system resources that are recorded in the journal. It is up to CICS users to develop this program.

CICS journals may also be used as audit trail files for system resources. Each CA Telon CICS journal definition is an independent object identified by the name of the journal. There are no sub-components.

Data administrators, designers, and programmers all use the CA Telon CICS journal definition to provide reusability of these objects within CA Telon file groups and programs.

File Group

The initial specification of a CICS journal access can be assisted by creating a file group in TELON option 2. This file group contains a list of all VSAM files, databases, DB2 tables, and CICS queues and journals that an application is likely to use. Instead of remembering the file names, you can simply enter auto exec and user exec statements under the line for the proper CICS journal.

You can bring a list of files, databases, tables, queues, and journals into your program definition in two ways:

- When creating the initial program definition, identify a BASE that already has a data group.
- From the Create/Update Data Group screen, enter the DGADD command and copy in file, database, or table information from a PSB, a file group, another program definition, a DBD definition, a VSAM file definition, a DB2 table definition, or a CICS queue or journal definition.

See the description of the Create/Update Data Group screen in the *Design Facility* Reference *Guide* for further explanation of the DGADD command.

Access Specification

You specify generated data access on the Create/Update Data Group screen for each CICS journal. Each journal is defined on a line of the Create/Update Data Group screen and identified by the characters CJNL=> in the first field on the line. A CICS journal line exists due to:

- The use of a BASE from another program definition
- A DGADD command
- The repeat or copy of another CJNL line in this program definition

The CICS journal line contains one additional field besides the CJNL=> indication: the name that identifies the journal to CA Telon. (CICS recognizes journals by the JOURNAL ID, a two-digit number affixed to the word "journal."; for example, JOURNAL04.) Note that the titles at the top of the screen refer to the auto exec and user exec lines, not to the CICS journal line. The CA Telon journal named on this line is used for all auto exec and user exec statements that are defined under it.

Information about the journal item follows the CJNL=> line. It is identified by the characters REC=> in the first field on the line and created at the same time as the CICS journal line. The information about the journal, such as COPY book name, can be modified from the Update CICS Journal Record screen. You access this screen by entering a **U** on the REC=> line.

You enter each auto exec and user exec statement on a line under the definition for the journal to be accessed. The general parameters for each access definition are described under "Generated Data Access" earlier in this chapter. Those parameters, which you enter on the Update CICS Journal Detail Data Access screen, cause the generation of appropriate EXEC CICS statements.

Write-Only

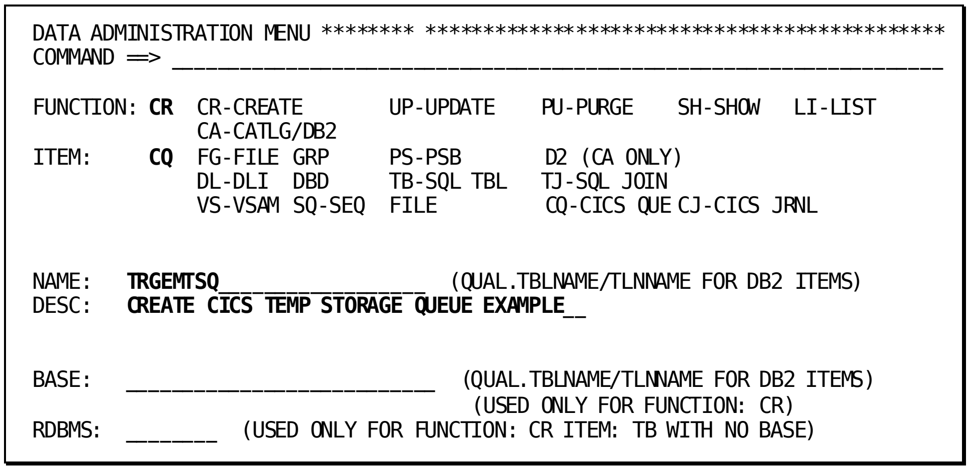
Since CICS journals can only be read from either COBOL or PL/I programs, CA Telon only generates an EXEC CICS JOURNAL data access request for CICS journals. This request can be either an auto exec or a user exec call.

WAIT JOURNAL VS. WAIT Option

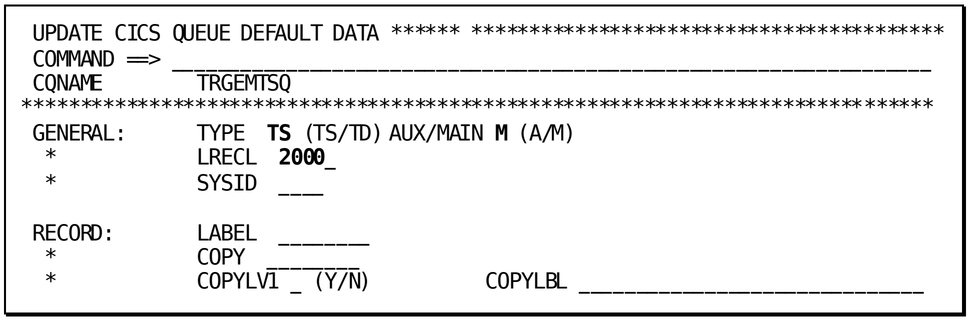
CA Telon allows you to choose between a generated EXEC CICS WAIT JOURNAL (specified by entering 'WAIT' in the FUNC field of the Update CICS Journal Detail Data Access screen) data access request, and an EXEC CICS JOURNAL data access request with the WAIT option (specified by entering **Y** in the WAIT field of the Update CICS Journal Detail Data Access screen). Consult IBM's CICS manuals for details on the differences between these two requests.

Creating CICS Queues

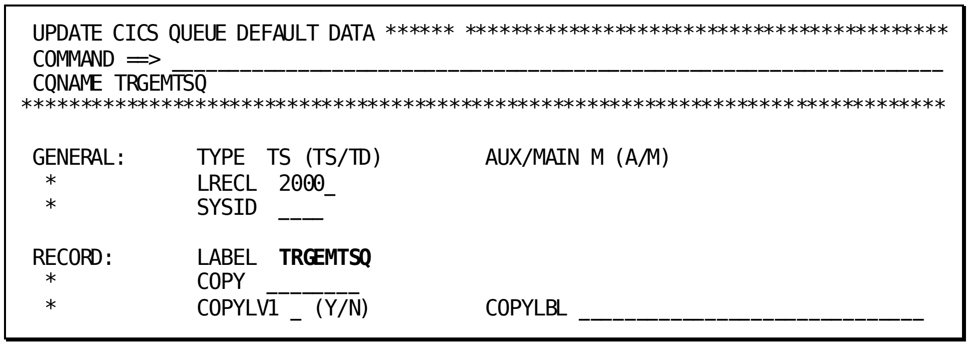
You can create CICS queues by accessing the Data Administration menu from the TDF Main menu. Type **CR** in the FUNCTION field, **CQ** in the ITEM field, the CICS queue name in the NAME field (in this case, **TRGEMTSQ**), a description in the DESC field, and press Enter.


```
DATA ADMINISTRATION MENU ******** ********************************************
COMMAND ==> ____________________________________________________________

FUNCTION: CR  CR-CREATE       UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
              CA-CATLG/DB2
ITEM:     CQ  FG-FILE GRP     PS-PSB       D2 (CA ONLY)
              DL-DLI  DBD     TB-SQL TBL   TJ-SQL JOIN
              VS-VSAM SQ-SEQ  FILE         CQ-CICS QUE CJ-CICS JRNL


NAME:    TRGEMTSQ_________________  (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    CREATE CICS TEMP STORAGE QUEUE EXAMPLE__


BASE:    __________________________  (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                      (USED ONLY FOR FUNCTION: CR)
RDBMS:   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update CICS Queue Default Data screen.


```
UPDATE CICS QUEUE DEFAULT DATA ****** ****************************************
COMMAND ==> ___________________________________________________________
CQNAME        TRGEMTSQ
*******************************************************************************
GENERAL:      TYPE  TS (TS/TD) AUX/MAIN M (A/M)
 *            LRECL  2000_
 *            SYSID  ____

RECORD:       LABEL  ________
 *            COPY  ________
 *            COPYLV1 _ (Y/N)         COPYLBL _____________________________
```

From the Update CICS Queue Default Data screen, type **TS** in the TYPE field (for temporary storage queues), **M** in the AUX/MAIN field (to indicate that the queue is written to main storage), the record length in the LRECL field (in this case, **2000**), and press Enter. CA Telon redisplays the Update CICS Queue Default Data screen. Type a label in the LABEL field (in this case, **TRGEMTSQ**). CA Telon uses this parameter to specify an override of the default I/O label for the CICS queue user exec data access request.



```
 UPDATE CICS QUEUE DEFAULT DATA ****** *****************************************
 COMMAND ==> ____________________________________________________________________
 CQNAME TRGEMTSQ
********************************************************************************

 GENERAL:      TYPE  TS (TS/TD)        AUX/MAIN M (A/M)
  *            LRECL  2000_
  *            SYSID  ____

 RECORD:       LABEL  TRGEMTSQ
  *            COPY   ________
  *            COPYLV1 _ (Y/N)         COPYLBL ______________________________
```

Press PF3 to update the information on this screen. CA Telon returns you to the Data Administration menu with the message **CICS QUEUE DEFAULT INFORMATION UPDATED** displayed in the upper right-hand corner.

Updating CICS Queues

You can update CICS queues by accessing the Data Administration menu from the TDF Main menu. Type **UP** in the FUNCTION field, **CQ** in the ITEM field, the CICS queue name in the NAME field (in this case, **TRGEMTSQ**), and press Enter.

```
DATA ADMINISTRATION MENU ************ *****CICS QUEUE DEFAULT INFORMATION UPDATED
COMMAND ==> _____________________________________________________________________



FUNCTION:UP    CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2

ITEM:       CQ         FG-FILE GRP      PS-PSB           D2 (CA ONLY)
                       DL-DLI DBD       TB-SQL TBL       TJ-SQL JOIN
                       VS-VSAM          SQ-SEQ FILE      CQ-CICS QUE
      CJ-CICS JRNL

NAME:    TRGEMTSQ__________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    _________________________________________


BASE:    ___________________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)

RDBMS:   ________   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update CICS Queue Default Data screen. From the Update CICS Queue Default Data screen, update the fields you choose. In this case, we are updating the COPY and COPYLBL fields.

```
UPDATE CICS QUEUE DEFAULT DATA **** ****************************************
COMMAND ==> _____________________________________________________________________
CQNAME TRGEMTSQ
*******************************************************************************

GENERAL:       TYPE    TS (TS/TD)       AUX/MAIN M (A/M)
  *            LRECL   2000
  *            SYSID   ____

RECORD:        LABEL   TRGEMTSQ
  *            COPY    EMPTSFMT
  *            COPYLV1 _ (Y/N)          COPYLBL EMPLOYEE-TS-IO-AREA____
```

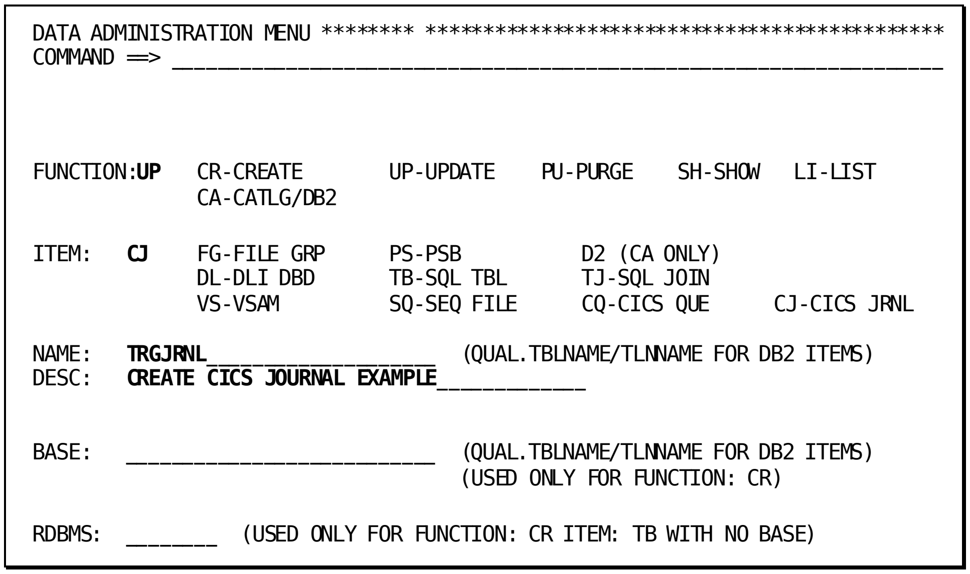
Type the copy name in the COPY field (in this case, **EMPTSFMT**). CA Telon uses this parameter to specify an override to the default COBOL COPY (or PL/I INCLUDE) member name that contains the queue record layout.

Type the copy label name in the COPYLBL field (in this case, **EMPLOYEE-TS-IO-AREA**). CA Telon uses this parameter to specify the COBOL (or PL/I) group level variable name for the CICS queue accesses.

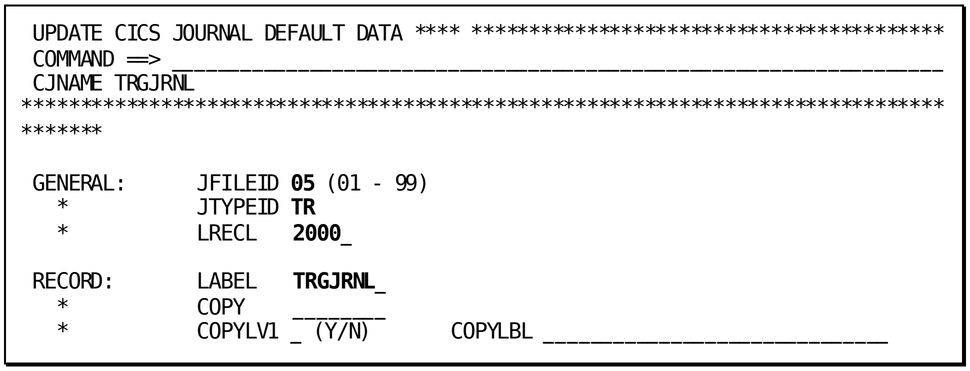
Press PF3 to update this screen. CA Telon returns you to the Data Administration menu with the message **CICS QUEUE DEFAULT INFORMATION UPDATED** in the upper right-hand corner.

Creating CICS Journals

You can create CICS journals by accessing the Data Administration menu from the TDF Main menu. Type **CR** in the FUNCTION field, **CJ** in the ITEM field, the CICS journal name in the NAME field (in this case, **TRGJRNL**), a description in the DESC field, and press Enter.


```
DATA ADMINISTRATION MENU ******** *********************************************
COMMAND ==> ___________________________________________________________________



FUNCTION:UP   CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
              CA-CATLG/DB2

ITEM:    CJ   FG-FILE GRP      PS-PSB           D2 (CA ONLY)
              DL-DLI DBD       TB-SQL TBL       TJ-SQL JOIN
              VS-VSAM          SQ-SEQ FILE      CQ-CICS QUE      CJ-CICS JRNL

NAME:    TRGJRNL___________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    CREATE CICS JOURNAL EXAMPLE_____________


BASE:    __________________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                      (USED ONLY FOR FUNCTION: CR)

RDBMS:   ________   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update CICS Journal Default Data screen.


```
 UPDATE CICS JOURNAL DEFAULT DATA **** *****************************************
 COMMAND ==> ___________________________________________________________________
 CJNAME TRGJRNL
*******************************************************************************
*******

 GENERAL:      JFILEID 05 (01 - 99)
   *           JTYPEID TR
   *           LRECL   2000_

 RECORD:       LABEL   TRGJRNL_
   *           COPY    ________
   *           COPYLV1 _ (Y/N)      COPYLBL ______________________________
```

From the Update CICS Journal Default Data screen, type a unique journal file ID (in this case, **05**). CA Telon uses this parameter in all generated I/O for this journal.

Type a journal type ID in the JTYPEID field (in this case, **TR**). CA Telon uses this parameter to indicate the two characters that are used by CICS to identify the origin of this journal record.

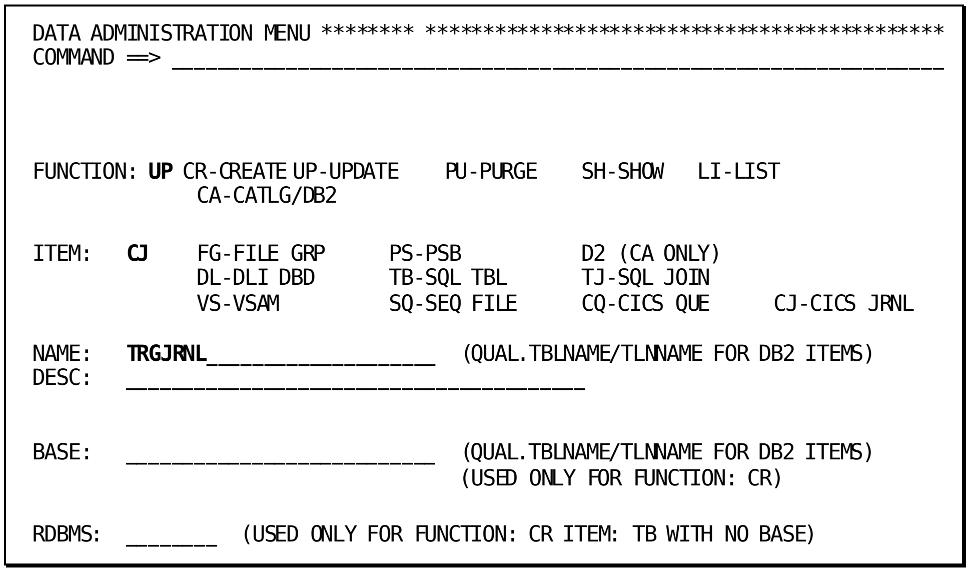
Type a record length in the LRECL field (in this case, **2000**).

Type a label in the LABEL field (in this case, **TRGJRNL**). CA Telon uses this label to specify a default to replace the CJNAME parameter in the user exec (that is, U-100) paragraph names that CA Telon generates for this journal.

Press PF3 to update the information on this screen. CA Telon returns you to the Data Administration menu with the message **CICS JOURNAL DEFAULT INFO UPDATED** displayed in the upper right-hand corner.

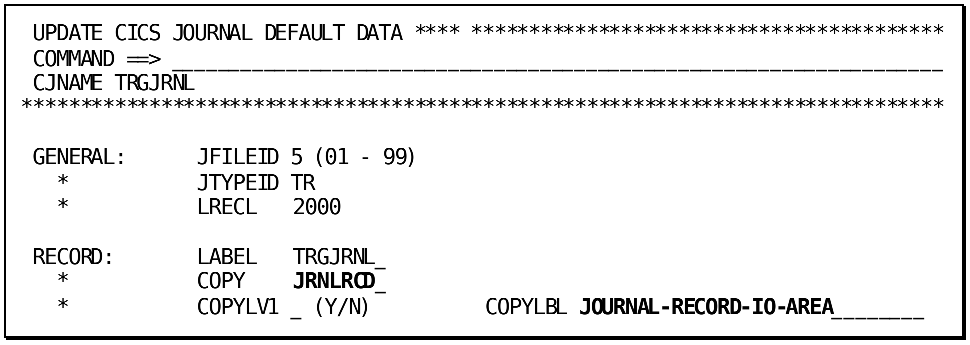
Updating CICS Journals

the TDF Main menu. Type **UP** in the FUNCTION field, **CJ** in the ITEM field, the CICS journal name in the NAME field (in this case, **TRGJRNL**), and press Enter.



```
DATA ADMINISTRATION MENU ******** *********************************************
COMMAND ==> __________________________________________________________________



FUNCTION: UP CR-CREATE UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
              CA-CATLG/DB2

ITEM:    CJ    FG-FILE GRP      PS-PSB          D2 (CA ONLY)
               DL-DLI DBD       TB-SQL TBL      TJ-SQL JOIN
               VS-VSAM          SQ-SEQ FILE     CQ-CICS QUE     CJ-CICS JRNL

NAME:    TRGJRNL___________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    ________________________________________

BASE:    ___________________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)

RDBMS:   ________   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update CICS Journal Default Data screen. From the Update CICS Journal Default Data screen, update the fields you choose. In this case, we are updating the COPY and COPYLBL fields.



```
 UPDATE CICS JOURNAL DEFAULT DATA **** ***************************************
 COMMAND ==> _________________________________________________________________
 CJNAME TRGJRNL
*****************************************************************************

 GENERAL:      JFILEID 5 (01 - 99)
   *           JTYPEID TR
   *           LRECL   2000

 RECORD:       LABEL   TRGJRNL_
   *           COPY    JRNLRCD_
   *           COPYLV1 _ (Y/N)         COPYLBL JOURNAL-RECORD-IO-AREA_______
```

Type the COPY name in the COPY field (in this case, **JRNLRCD**). CA Telon uses this parameter to specify an override to the default COBOL COPY (or PL/I INCLUDE) member name that contains the journal record layout.

Type the COPY label name in the COPYLBL field (in this case, **JOURNAL-RECORD-IO-AREA**). CA Telon uses this parameter to specify the COBOL (or PL/I) group level variable name for the CICS journal accesses.

Press PF3 to update this screen. CA Telon returns you to the Data Administration menu with the message **CICS QUEUE DEFAULT INFORMATION UPDATED** in the upper right-hand corner.

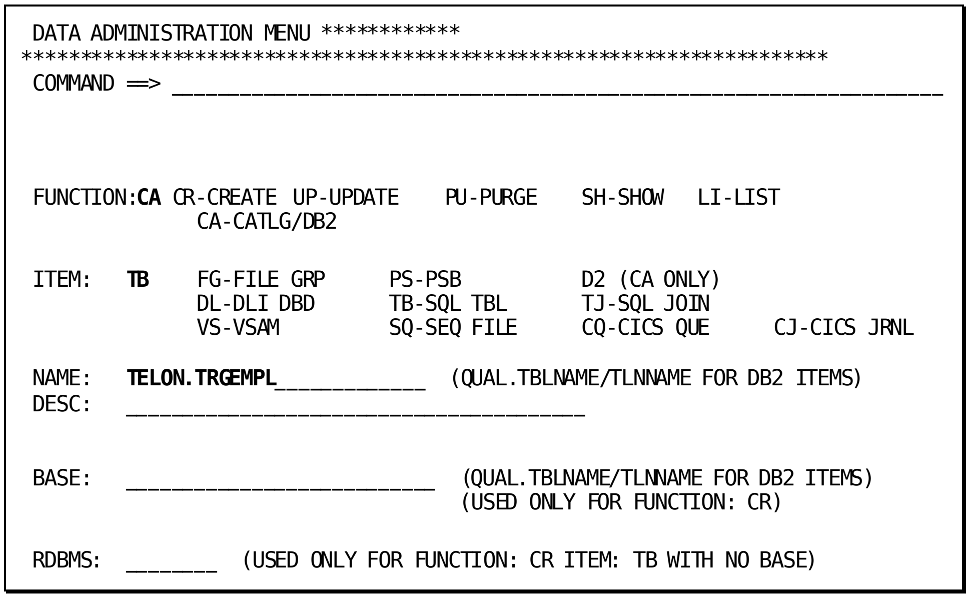
Chapter 7: DB2 Tasks

CA Telon makes the task of importing TABLE definitions from a DB2 catalog online simple through a series of easy-to-use screens.

Processing on CA Telon's screens enables you to bring information from the DB2 catalog into CA Telon. This allows you to view DB2 through CA Telon's eyes. In other words, CA Telon gives you an easy way to access information stored on DB2. All information needed to generate SQL statements to access a DB2 table as a result of an auto exec or user exec request must reside in the CA Telon directory.

Importing TABLE Definitions Online

Access the TDF Main menu by the standards set up at your installation. From this Main menu, type **2** in the FUNCTION field and press Enter. This accesses the Data Administration menu.



```
 DATA ADMINISTRATION MENU ************
*********************************************************************
 COMMAND ==> _________________________________________________________________



 FUNCTION:CA CR-CREATE  UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
               CA-CATLG/DB2

 ITEM:    TB     FG-FILE GRP      PS-PSB          D2 (CA ONLY)
               DL-DLI DBD       TB-SQL TBL      TJ-SQL JOIN
               VS-VSAM          SQ-SEQ FILE     CQ-CICS QUE      CJ-CICS JRNL

 NAME:    TELON.TRGEMPL_____________  (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
 DESC:    _______________________________________

 BASE:    __________________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)

 RDBMS:   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

From this menu, type **CA** in the FUNCTION field, **TB** in the ITEM field, the name of the DB2 table in the NAME field (in this case, **TELON.TRGEMPL**) and press Enter. This accesses the Catalog/Import DB2 Tables screen. This screen displays a list of tables in the DB2 catalog beginning with TELON.TRGEMPL. PF8 and PF7 allow you to page forward and backward.

```
 CATALOG/IMPORT DB2 TABLES ***********
*******************************************************************
 COMMAND ==>                              PAGE 01 MORE

 SELECT THE DB2 TABLES TO BE IMPORTED INTO THE TDF
  **QUAL** ****TABLENAME***** COLS TLN TLNNAME   USERID  UPDATE
  TELON   TRGEMPL        13   Y   TRGEMPL  ROSENH  09/15/2000
  TELON   TRGEMPLA       13   Y   TRGEMPLA         NARTKER 04/27/2000
  TELON   TRGEMPLB       13   Y   TRGEMPLQ         QUINN   06/14/2000
  TELON   TRGEMPLC       13   N
  TELON   TRGEMPLD       13   N
  TELON   TRGEMPLE                13  N
  TELON   TRGEMPLF       13   N
  TELON   TRGEMPL0       13   N
  TELON   TRGEMPL1       13   Y   TRGEMPL1         MOSSM   01/17/2001
  TELON   TRGEMPL2       13   Y   TRGEMPL2         ANDERST 10/02/2000
  TELON   TRGEMPL3       13   Y   TRGEMPL3         SIMONS  03/14/2000
  TELON   TRGEMPL4       13   Y   TRGEMPL4         MOSSM   06/27/2000
  TELON   TRGEMPL5       13   N
  TELON   TRGEMPL6       13   Y   TRGEMPL6         SIMONS  10/20/2000
  TELON   TRGEMPL7       13   N
  TELON   TRGEMPL8       13   N
  TELON   TRGEMPL9       13   Y   TRGEMPL9         DELLI   10/17/1999
  TELON   TRGTASK         9   N
  TELON   TRGTASKA        9   N
```

Note that CA is the function for listing or importing DB2 tables. CA is only used for DB2-related processing on OS/390 and z/OS. This function is not available on CA Telon PWS.

The Catalog/Import DB2 Tables screen shows a list of DB2 tables from which you can import or select the tables you need.

Since every DB2 table has two components, a qualifier and table name, CA Telon displays these in the first two columns (QUAL and TABLENAME columns) of the Catalog/Import DB2 Tables screen. CA Telon gets the TABLENAME names from the DB2 catalog.

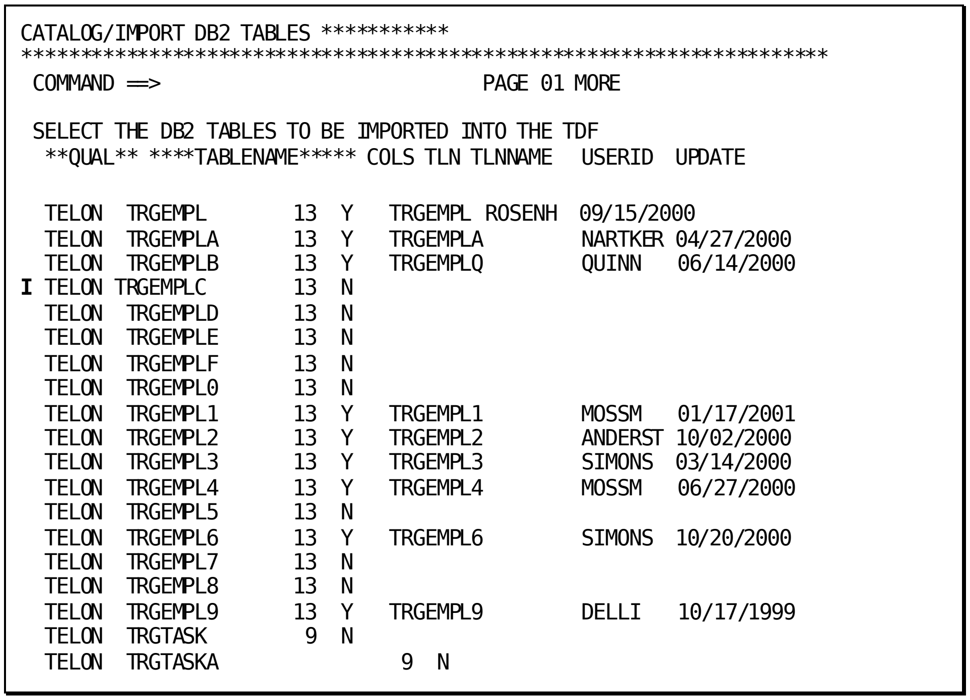
The COLS column shows the number of columns in the table and this is also obtained from the DB2 catalog.

The TLN column indicates whether or not the table has been imported into the CA Telon directory before.

The TLNNAME column specifies a mnemonic name for the DB2 table or join. The table can be referred to by its TLNNAME in CA Telon without specifying the fully qualified table name. This is the *only* parameter you can alter on this screen.

The USERID column indicates the user that last imported the table. The UPDATE column indicates the date of the *first* update or import.

You can import the TELON.TRGEMPLC DB2 table by entering **I** in the left-hand column and pressing Enter.



```
CATALOG/IMPORT DB2 TABLES ***********
**********************************************************************
 COMMAND ==>                            PAGE 01 MORE

 SELECT THE DB2 TABLES TO BE IMPORTED INTO THE TDF
  **QUAL** ****TABLENAME***** COLS TLN TLNNAME   USERID  UPDATE

  TELON  TRGEMPL       13  Y   TRGEMPL ROSENH  09/15/2000
  TELON  TRGEMPLA      13  Y   TRGEMPLA        NARTKER 04/27/2000
  TELON  TRGEMPLB      13  Y   TRGEMPLQ        QUINN   06/14/2000
I TELON TRGEMPLC       13  N
  TELON  TRGEMPLD      13  N
  TELON  TRGEMPLE      13  N
  TELON  TRGEMPLF      13  N
  TELON  TRGEMPL0      13  N
  TELON  TRGEMPL1      13  Y   TRGEMPL1        MOSSM   01/17/2001
  TELON  TRGEMPL2      13  Y   TRGEMPL2        ANDERST 10/02/2000
  TELON  TRGEMPL3      13  Y   TRGEMPL3        SIMONS  03/14/2000
  TELON  TRGEMPL4      13  Y   TRGEMPL4        MOSSM   06/27/2000
  TELON  TRGEMPL5      13  N
  TELON  TRGEMPL6      13  Y   TRGEMPL6        SIMONS  10/20/2000
  TELON  TRGEMPL7      13  N
  TELON  TRGEMPL8      13  N
  TELON  TRGEMPL9      13  Y   TRGEMPL9        DELLI   10/17/1999
  TELON  TRGTASK        9  N
  TELON  TRGTASKA               9  N
```

You can import one or more tables. Other valid values are "O" if you want to overlay the information in CA Telon about a table with the current table definition in the DB2 catalog. A value of "A" adds new information from the DB2 catalog to what is already in CA Telon, but does not overlay any information.

When you press Enter, the screen refreshes with TELON displaying TRGEMPLC as the TLNNAME and giving the **IMPORT OK** message to show that the import was successful. If you do not like the default TLNNAME provided by CA Telon, then you can change it.

You have completed the import of your DB2 table. Press PF3 to return to the Data Administration screen and to end processing on this screen.

Batch DB2 Catalog Extract

The Batch DB2 Catalog Extract is the facility used to extract SQL table information from the DB2 catalog and build a CA Telon Transport file to load the development platform TDF with the table information.

The facility additionally provides the ability to produce DCL and copy members for inclusion in the application programs.

To operate this system, a control file must be built by a text editor. Once this control file is prepared, the batch driver program is invoked and the Transport file and copy members are created from the tables identified in the control file.

Components of the Batch DB2 Catalog Extract Structure

The Batch DB2 Catalog Extract application structure is comprised of the components described below.

BATCH DRIVER (CM10/PM10)

This program lets you specify which tables will be extracted from the DB2 system in batch mode. The CM10 program operates in COBOL, and the PM10 program in PL/I.

The batch driver takes a series of input transactions specifying tables to be selected, userids and passwords, and optionally information concerning DCL and copy members.

Once the input information is processed, the CM20 (or PM20) program is called to build the Transport file and the copy members. There is a series of 80 character input transactions loaded in the file SQINTRAN, which serves as input to the CM10 or PM10 program:

Columns	Description
1-8	Transaction ID, with the following values: ■ USERID ■ TABLE ■ TABLEDCL ■ TABLECPY **Note:** The USERID transaction ID is specified once, whereas there may be multiple TABLEs (followed optionally by TABLEDCL and/or TABLECPY for each table).
9-80	Data for specific Transaction IDs

For the USERID Transaction ID, the columns 9-23 contain the userid value.

For the TABLE transaction ID, the columns 9-80 have the following usage:

Columns	Description
9-38	Table name
39-46	TLNNAME
47	Create DCL (Y/N)?
48	Create copy member (Y/N)?
49	DCL redefine (Y/N)
50	Synonym (Y/N)
51-80	Table owner

For the TABLEDCL transaction ID, the columns 9-46 have the following usage:

Columns	Description
9-38	DCLLBL
39-46	DCL member name (if different from TLNNAME)

For the TABLECPY transaction ID, the columns 9-47 have the following usage:

Columns	Description
9-38	COPYLBL
39-46	Copy member name (if different from TLNNAME)
47	COPYLV1 (Y/N)

BUILD TRANSPORT (CM20/PM20)

This batch subroutine is called by the batch driver CM10 (or PM10). A Transport file is built for all selected tables and a report is produced summarizing the results.

For each table, if either DCL or copy members are to be created, a call is made to the CM30 (or PM30) program.

When the Transport file and copy members are built, the processing ends.

BUILD COPY MEMBERS (CM30/PM30)

This batch subroutine is called by CM20 (or PM20) to produce either DCL or copy members.

Building of the Batch DB2 Catalog Extract

The Batch DB2 Catalog Extract is supplied as load modules in the CA Telon installation, but it may also be built from CA Telon source code supplied with the product. For the mainframe, there is JCL provided to build this system from CA Telon source code and copy members (found in the SOURCE dataset), or from fully resolved generated code.

On the PWS, the CA Telon source code and copy members are found in the DB2EXT subdirectory. In addition, the DB2EXT subdirectory contains JCL and procedures which need to be uploaded to the mainframe to build the system, once the generate and resolve steps have been executed on the PWS using the GENERATE command.

Member	**Type**	**Use**
SQD2CMPC	JCL	For building the Batch DB2 Catalog Extract system from fully resolved COBOL code which has been uploaded from the PWS.
SQD2CMPP	JCL	For building the Batch DB2 Catalog Extract system from fully resolved PL/I code which has been uploaded from the PWS.
SQD2GENC	JCL	For building the Batch DB2 Catalog Extract system from CA Telon source code for COBOL on the mainframe.
SQD2GENP	JCL	For building the Batch DB2 Catalog Extract system from CA Telon source code for PL/I on the mainframe.
SQ2BCL	Procedure	Invoked by SQD2CMPC to build the Batch DB2 Catalog Extract system from fully resolved COBOL code uploaded from the PWS.
SQ2BPL	Procedure	Invoked by SQD2CMPP to build the Batch DB2 Catalog Extract system from fully resolved PL/I code uploaded from the PWS.
SQ2BPGCL	Procedure	Invoked by SQD2GENC to build the Batch DB2 Catalog Extract system from CA Telon source code for COBOL on the mainframe.

Member	Type	Use
SQ2BPGPL	Procedure	Invoked by SQD2GENP to build the Batch DB2 Catalog Extract system from CA Telon source code for PL/I on the mainframe.

Within the SOURCE dataset (the DB2EXT subdirectory on the PWS) may be found the CA Telon source code and copy members for generation and resolution of the Batch DB2 Catalog Extract system, along with cards for the link step, as follows:

Member	Description
SQCM10BD	CA Telon source for COBOL program CM10
SQCM10LK	Link cards for CM10 program
SQCM20BD	CA Telon source for COBOL program CM20
SQCM20LK	Link cards for CM20 program
SQCM30BD	CA Telon source for COBOL program CM30
SQCM30LK	Link cards for CM30 program
SQPM10BD	CA Telon source for PL/I program PM10
SQPM10LK	Link cards for PM10 program
SQPM20BD	CA Telon source for PL/I program PM20
SQPM20LK	Link cards for PM20 program
SQPM30BD	CA Telon source for PL/I program PM30
SQPM30LK	Link cards for PM30 program
SQSTDD2C	Copy member for COBOL programs
SQSTDD2P	Copy member for PL/I programs
SQXFERC	Copy member for COBOL programs
SQXFERC1	Copy member for COBOL programs
SQXFERC2	Copy member for COBOL programs
SQXFERP	Copy member for PL/I programs
SQXFERP1	Copy member for PL/I programs
SQXFERP2	Copy member for PL/I programs
SYSCOLSC	Copy member for COBOL programs
SYSCOLSP	Copy member for PL/I programs
SYSTBLSC	Copy member for COBOL programs

Member	Description
SYSTBLSP	Copy member for PL/I programs

Execution of the Batch DB2 Catalog Extract

There are two JCL decks provided which are used to operate the Batch DB2 Catalog Extract:

- SQD2EXTC-for COBOL operation
- SQD2EXTP-for PL/I operation

Note: The Batch DB2 Catalog Extract must be run on the mainframe,even for PWS installations. The resulting Transport file must then be downloaded to the PWS for importing into the TDF. The copy members may also be made available for the resolve step in the GENERATE command for application programs.

Batch DB2 Stored Procedure Catalog Extract

The Batch DB2 Stored Procedure Catalog Extract is the facility used to extract stored procedure information from the DB2 catalog and build a CA Telon Transport file to load the development platform TDF with the stored procedure information.

To operate this system, a control file must be built by a text editor. Once this control file is prepared, the batch driver program is invoked and the Transport file is created from the stored procedures identified in the control file.

Components of the Batch DB2 Stored Procedure Catalog Extract Structure

The Batch DB2 Stored Procedure Catalog Extract application structure is comprised of the components described below.

BATCH DRIVER (CM10/PM10)

This program lets you specify which stored procedures will be extracted from the DB2 system in batch mode. The CM10 program operates in COBOL, and the PM10 program in PL/I.

The batch driver takes a series of input transactions specifying stored procedures to be selected, as well as a userid, Once the input information is processed, the CM20 (or PM20) program is called to build the Transport file containing information extracted from the DB2 catalog. There are a series of 80 character input transactions loaded in the file SPINTRAN, which serves as input to the CM10 or PM10 program:

There are two valid transaction ID's, specified in columns 1-8: STORED, and USERID. The USERID transaction ID is specified once, whereas there may be multiple STORED records.

For the USERID transaction ID, the userid is specified in columns 10-17.

For the STORED transaction ID, the columns 10-41 have the following usage:

Columns	Description
10-17	Schema
18-35	Stored procedure name
36-41	Header-ID

Note: the Header-ID specification is required, since there is no certainty that the stored procedure name found in the DB2 catalog will follow the CA Telon entity naming convention.

BUILD TRANSPORT (CM20/PM20)

This batch subroutine is called by the batch driver CM10 (or PM10). A Transport file is built for all selected stored procedures, and a report is produced summarizing the results. When the Transport file is built, the processing ends.

Building of the Batch DB2 Stored Procedure Catalog Extract

The Batch DB2 Stored Procedure Catalog Extract is supplied as load modules in the CA Telon installation, but it may also be built from CA Telon source code supplied with the product. For the mainframe, there is JCL provided to build this system from CA Telon source code and copy members (found in the SOURCE dataset), or from fully resolved generated code.

On the PWS, the CA Telon source code and copy members are found in the DB2EXT subdirectory. In addition, the DB2EXT subdirectory contains JCL and procedures that need to be uploaded to the mainframe to build the system, once the generate and resolve steps have been executed on the PWS.

Member	Type	Use
SQD2CMPC	JCL	For building the Batch DB2 Stored Procedure Catalog Extract system from fully resolved COBOL code which has been uploaded from the PWS.

Member	Type	Use
SQD2CMPP	JCL	For building the Batch DB2 Stored Procedure Catalog Extract system from fully resolved PL/I code which has been uploaded from the PWS.
SPD2GENC	JCL	For building the Batch DB2 Stored Procedure Catalog Extract system from CA Telon source code for COBOL on the mainframe.
SPD2GENP	JCL	For building the Batch DB2 Stored Procedure Catalog Extract system from CA Telon source code for PL/I on the mainframe.
SQ2BCL	Procedure	Invoked by SQD2CMPC to build the Batch DB2 Stored Procedure Catalog Extract system from fully resolved COBOL code uploaded from the PWS.
SQ2BPL	Procedure	Invoked by SQD2CMPP to build the Batch DB2 Stored Procedure Catalog Extract system from fully resolved PL/I code uploaded from the PWS.
SQ2BPGCL	Procedure	Invoked by SPD2GENP to build the Batch DB2 Stored Procedure Catalog Extract system from CA Telon source code for PL/I on the mainframe.
SQ2BPGPL	Procedure	

Within the SOURCE dataset (the DB2EXT subdirectory on the PWS) may be found the CA Telon source code and copy members for generation and resolution of the Batch DB2 Stored Procedure Catalog Extract system, along with cards for the link step, as follows:

Member	Description
SPCM10BD	CA Telon source for COBOL program CM10
SPCM10LK	Link cards for CM10 program
SPCM20BD	CA Telon source for COBOL program CM20
SPCM20LK	Link cards for CM20 program
SPPM10BD	CA Telon source for PL/I program PM10
SPPM10LK	Link cards for PM10 program
SPPM20BD	CA Telon source for PL/I program PM20

Member	Description
SPPM20LK	Link cards for PM20 program
SPSTDD2C	Copy member for COBOL programs
SPSTDD2P	Copy member for PL/I programs
SPXFERC	Copy member for COBOL programs
SPXFERP	Copy member for PL/I programs
SYSROUTC	Copy member for COBOL programs
SYSROUTP	Copy member for PL/I programs
SYSPARMC	Copy member for COBOL programs
SYSPARMP	Copy member for PL/I programs

Execution of the Batch DB2 Stored Procedure Catalog Extract

There are two JCL decks provided which are used to operate the Batch DB2 Stored Procedure Catalog Extract:

- SQD2EXTC for COBOL operation
- SQD2EXTP for PL/I operation

Note: The Batch DB2 Stored Procedure Catalog Extract must be run on the mainframe, even for PWS installations. The resulting Transport file must then be downloaded to the PWS for importing into the TDF.

Chapter 8: Setting Up File Groups

File groups (FGs) enable you to group together data sets, databases, and tables. File groups speed up your processing by gathering together all the logical pieces of your particular project.

Without file groups you can still use CA-Telon, but you will have to process each access to a data set, database, and table. File groups enable you to combine this processing in one step. With one command you can access completely different databases, data sets, and tables.

This chapter shows you how to set up file groups (FGs).

The first subject, "The File Group Concept," describes important concepts pertinent to setting up file groups. The later subjects give sample sessions for creating and updating file groups.

File Group Concept

A *CA Telon file group* is a collection of data objects that can be used as the DBMS. File groups can contain references to objects constructed using Information Management System (IMS), Structured Query Language (SQL), Customer Information Control System (CICS), Virtual Sequential Access Method (VSAM), or sequential data set methods. If several programs have similar data access requirements and they each access multiple environments, defining a file group can help keep the program's data access synchronized during development and maintenance.

Note that while most objects defined to Data Administration correspond to objects in a DBMS, the CA Telon file group has no precedent in existing DBMSs. It is a special construct that is local to CA Telon and was created for the convenience of CA Telon users. Like a CA Telon IMS PSB, it allows groups of data objects to be gathered into one entity. Construction of CA Telon file groups is optional.

A CA Telon file group consists of references to one or more of the following:

- IMS Program Communications Block (PCB)
- DB2 SQL table
- Data set
- CICS queue
- CICS journal

An IMS PCB is a structure used for communicating with the IMS system and IMS databases. There are three types of PCBs that may be defined in a file group. They are:

- Teleprocessing PCBs (TP-PCB)
- Database PCBs (DB-PCB)
- GSAM PCBs (GSAM-PCB)

The first, a *teleprocessing PCB (TP-PCB),* is used for sending messages through the IMS data communications teleprocessing monitor. The second, a *database PCB (DB-PCB)*, is used for storing and retrieving information from IMS databases. Database PCBs are further composed of one or more *sensitive segments,* that define the segments in the database that this PCB can access. Database PCBs must contain at least one sensitive segment. The third type of PCB, a *generalized sequential access method PCB (GSAM-PCB)*, allows standard sequential data sets to be manipulated using IMS calls.

DB2 SQL tables are collections of data defined using SQL in terms of rows and columns. A special type of table, the CA Telon *joined table,* allows several related tables to be treated as one large table (DB2 SQL tables and CA Telon joined tables are defined in detail in other chapters of this guide). DB2 SQL tables and CA Telon joined tables are composed of one or more CA Telon rows and their columns subsets.

A *CA Telon row* is a row name that is unique within a CA Telon DB2 SQL table definition. A CA Telon row consists of a subset of columns that are associated only with the row. A *column subset* contains one or more DB2 SQL columns, and can combine columns in the table. Identical columns can exist in different CA Telon rows. (A CA Telon row is not a DB2 SQL table row.) A CA Telon DB2 SQL table must contain at least one CA Telon row. CA Telon uses the CA Telon row name in the generated I/O procedure or data access requests (paragraph names) in program generation.

File groups can also contain references to two types of data sets: *sequential data sets* and *VSAM data sets,* which are data sets created independent of any particular DBMS.

A *CICS queue* is a file created by CICS for scratchpad or inter-program communication purposes. A *CICS journal* is a chronological record of all changes made to CICS resources.

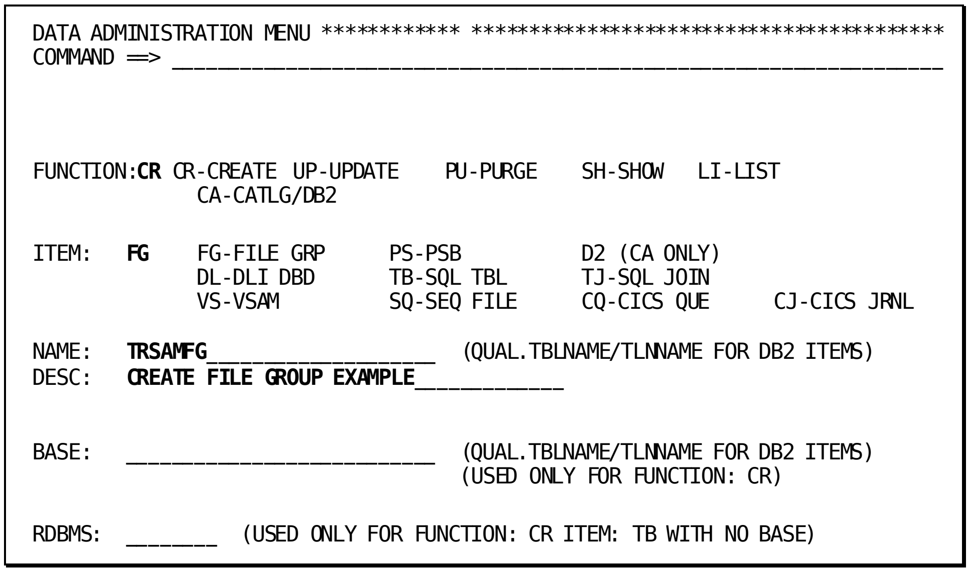
File groups are created in Data Administration option 2. They consist of one or more file, database, PSB or DB2 table definitions in any combination. File groups are useful when creating data groups for programs that will access multiple data objects.

For example, if one program must access a VSAM file, a DL/I database and a DB2 table, a file group can be created in option 2 that contains all three objects. In the data group for the program, a single DGADD command will be all that is needed to retrieve the three data objects from Data Administration and add them to the data group. Some programs access large numbers of files, databases, and other data objects.

Any data object that can be defined in Data Administration is eligible to be included in a file group.

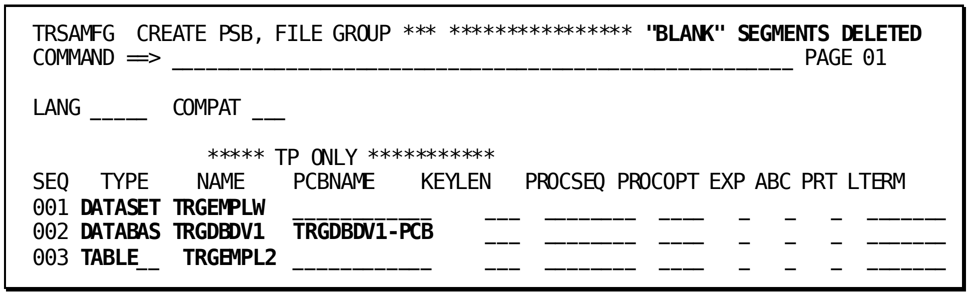
Creating File Groups

You can create file groups by accessing the Data Administration menu from the TDF Main menu. Type **CR** in the FUNCTION field, **FG** in the ITEM field, the file group Name in the NAME field (in this case, **TRSAMFG**), a description in the DESC field, and press Enter.



```
DATA ADMINISTRATION MENU ************ *****************************************
COMMAND ==> ____________________________________________________________________



FUNCTION:CR CR-CREATE  UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
              CA-CATLG/DB2

ITEM:    FG    FG-FILE GRP      PS-PSB          D2 (CA ONLY)
               DL-DLI DBD       TB-SQL TBL      TJ-SQL JOIN
               VS-VSAM          SQ-SEQ FILE     CQ-CICS QUE      CJ-CICS JRNL

NAME:    TRSAMFG____________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    CREATE FILE GROUP EXAMPLE_____________


BASE:    ___________________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)

RDBMS:   ________  (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Create PSB, File Group screen.



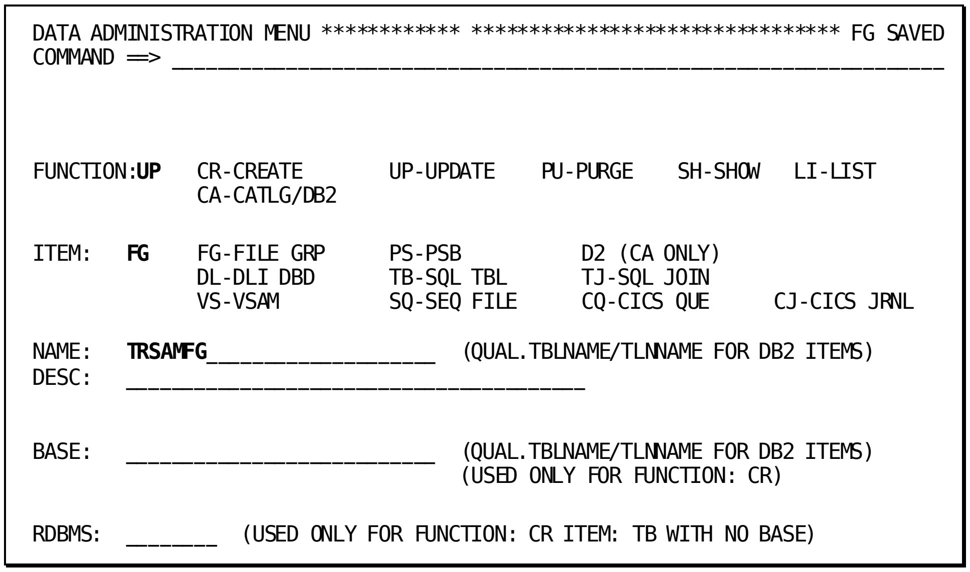
```
TRSAMFG  CREATE PSB, FILE GROUP *** **************** "BLANK" SEGMENTS DELETED
COMMAND ==> _______________________________________________________ PAGE 01

LANG _____   COMPAT ___

                ***** TP ONLY ***********
SEQ  TYPE    NAME     PCBNAME    KEYLEN  PROCSEQ PROCOPT EXP ABC PRT LTERM
001 DATASET TRGEMPLW  ____________    ___  ________ ____   _   _   _  ______
002 DATABAS TRGDBDV1  TRGDBDV1-PCB    ___  ________ ____   _   _   _  ______
003 TABLE__  TRGEMPL2 ____________    ___  ________ ____   _   _   _  ______
```

From the Create PSB, File Group screen you can create a file group. This example shows a file group containing a data set called **TRGEMPLW**, a database called **TRGDBDV1**, and a table called **TRGEMPL2**. The database has a PCB Name called **TRGDBDV1**. Input the information as shown on this screen and press Enter.

Press PF3 to update the information. You return to the Data Administration menu. CA Telon saves the file group you created and displays the message **FG SAVED** in the upper right-hand corner.

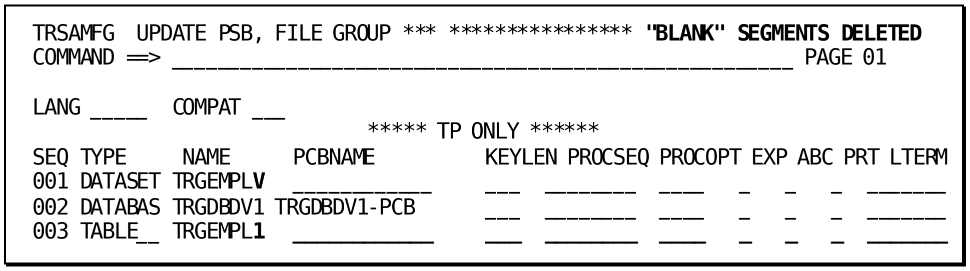
Updating File Groups

You can update file groups by accessing the Data Administration menu from the TDF Main menu. Type in the FUNCTION field, **FG** in the ITEM field, the file group name in the NAME field (in this case, **TRSAMFG**), and press Enter.



```
DATA ADMINISTRATION MENU ************ ******************************** FG SAVED
COMMAND ==> ___________________________________________________________________



FUNCTION:UP   CR-CREATE        UP-UPDATE    PU-PURGE    SH-SHOW   LI-LIST
              CA-CATLG/DB2

ITEM:    FG   FG-FILE GRP      PS-PSB           D2 (CA ONLY)
              DL-DLI DBD       TB-SQL TBL       TJ-SQL JOIN
              VS-VSAM          SQ-SEQ FILE      CQ-CICS QUE      CJ-CICS JRNL

NAME:    TRSAMFG____________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
DESC:    ________________________________________

BASE:    ___________________________   (QUAL.TBLNAME/TLNNAME FOR DB2 ITEMS)
                                       (USED ONLY FOR FUNCTION: CR)

RDBMS:   ________   (USED ONLY FOR FUNCTION: CR ITEM: TB WITH NO BASE)
```

CA Telon displays the Update PSB, File Group screen. From the Update PSB, File Group screen, update the fields you choose. In this case, we are updating the data set and table names in the file group.



```
TRSAMFG  UPDATE PSB, FILE GROUP *** **************** "BLANK" SEGMENTS DELETED
COMMAND ==> ______________________________________________________ PAGE 01

LANG _____   COMPAT ___
                                 ***** TP ONLY ******
SEQ TYPE     NAME     PCBNAME          KEYLEN PROCSEQ PROCOPT EXP ABC PRT LTERM
001 DATASET TRGEMPLV  ____________     ___   ________ ____     _   _   _  _______
002 DATABAS TRGDBDV1 TRGDBDV1-PCB      ___   ________ ____     _   _   _  _______
003 TABLE__ TRGEMPL1  ____________     ___   ________ ____     _   _   _  _______
```

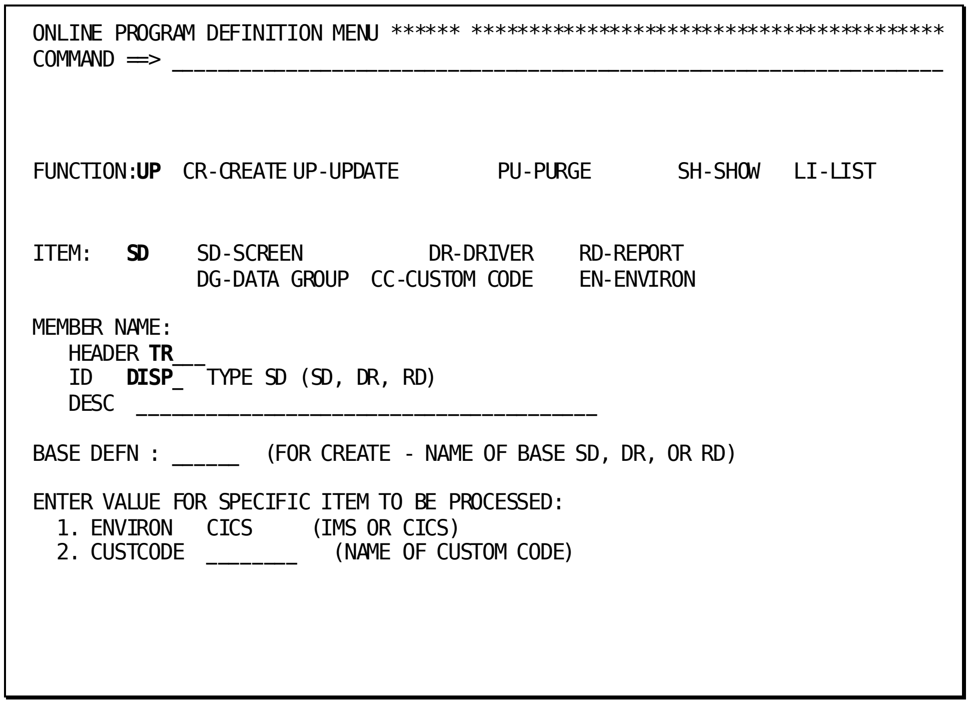
This example shows the TRGEMPLW data set being changed to TRGEMPL**V** and the TRGEMPL2 data set being changed to TRGEMPL**1**.

Press Enter, then press PF3 to update the file group changes. CA Telon then returns you to the Data Administration menu and displays the message **FG SAVED** in the upper right-hand corner.

At this point, you can type **=4** on the COMMAND line to transfer to the Online Program Definition menu.

Updating the Screen Definition

From the Online Program Definition menu, type in the FUNCTION field, **SD** in the ITEM field, the appropriate HEADER and ID (in this case, **TR** and **DISP**), and press Enter.



```
ONLINE PROGRAM DEFINITION MENU ****** ****************************************
COMMAND ==> __________________________________________________________________



FUNCTION:UP  CR-CREATE UP-UPDATE       PU-PURGE      SH-SHOW   LI-LIST


ITEM:   SD    SD-SCREEN          DR-DRIVER    RD-REPORT
              DG-DATA GROUP  CC-CUSTOM CODE    EN-ENVIRON

MEMBER NAME:
   HEADER TR__
   ID   DISP_  TYPE SD (SD, DR, RD)
   DESC  ________________________________________

BASE DEFN : _____  (FOR CREATE - NAME OF BASE SD, DR, OR RD)

ENTER VALUE FOR SPECIFIC ITEM TO BE PROCESSED:
  1. ENVIRON   CICS     (IMS OR CICS)
  2. CUSTCODE  _______    (NAME OF CUSTOM CODE)
```

After you press Enter, CA Telon displays the Update Screen Definition screen for the member TRDISP.SD. Type a **U** in the DATA GROUP field and press Enter to update the data group.

```
TRDISP.SD UPDATE SCREEN DEFINITION ** *****************************************
 COMMAND ==> ___________________________________________________________________
 OPTIONS ==> CUSTOM CODE _ DATA GROUP U PANEL DEF _ ENV CICS _ SCRN PARMS _

 GENERAL:      DESC    CICS EMPLOYEE DISPLAY________________  _ REMARKS
**DFLT**
   *           NEXTPGM _____ CURSOR NEWDISP_ SIZE 24 X 80   LANG COB (COB/PLI)

 DATA          XFERWKA ______________________________________________________
 AREAS:   _    WKAREA  ______________________________________________________

 OUTPUT:
 A-100 _       OINIT1  _______    _ OINIT2 _______    _ CURSCUS _______
 B-100 _       OUTTERM _______

 INPUT:
 P-100         PFKEYS  ______________________________________________________
 D-100 _       ININIT1 _______    _ ININIT2 _______
 J-100 _       SELECT FIELDS
 E-100 _       FLDEDIT _______
 X-100 _       SCREEN XFEDIT/SEGEDIT _ CONSIS _______
 H-100 _       INTERM _______

 MISC: _       SECTION______________________________________________________
   *           PGMCUST______________________________________________________
```

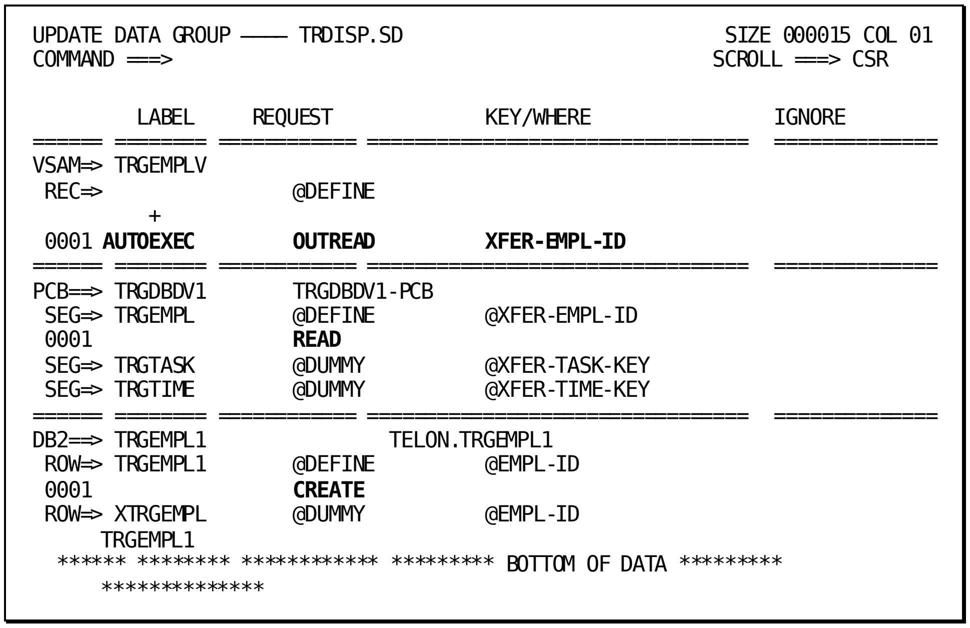
CA Telon displays the Update Data Group screen. From the Update Data Group screen, type the appropriate DGADD command (in this case, **DGADD TRSAMFG**) and press Enter.

```
 UPDATE DATA GROUP ---- TRDISP.SD   ** USE "DGADD" COMMAND TO INIT DATA GROUP
 COMMAND ===> DGADD TRSAMFG                 SCROLL ===> CSR
        LABEL    REQUEST                KEY/WHERE                 IGNORE
 ===== ======= =========== ============================= ===========
  ****** ******** ************ ********* BOTTOM OF DATA ******** *************
```

CA Telon redisplays the Update Data Group screen with the requested information. The message **DATA GROUP SUCCESSFULLY EXPANDED** displays in the upper right corner.

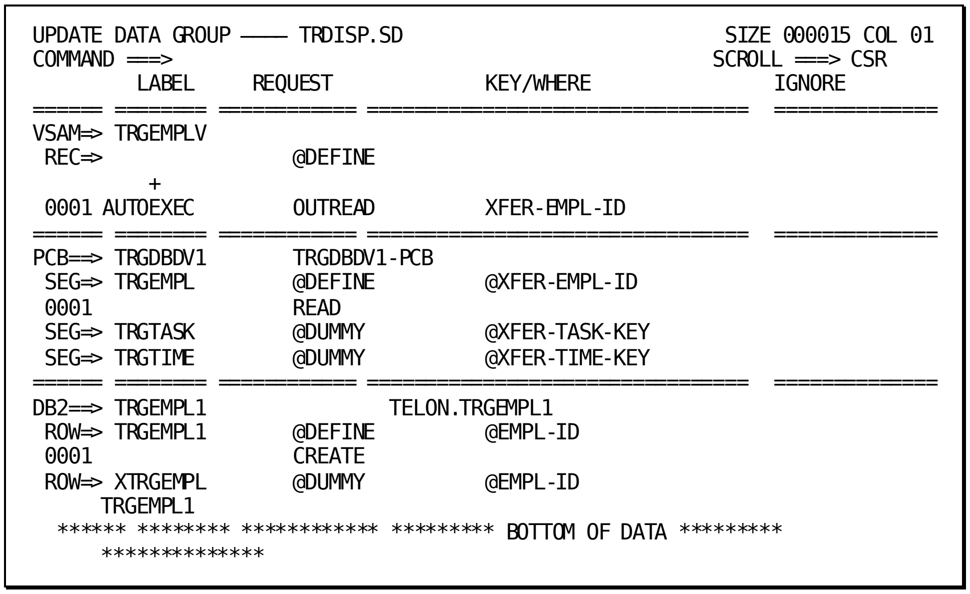
For a complete list of DGADD commands available on this screen, refer to the Design Facility *Reference Guide*.

From the Update Data Group screen, use the line command **I** (Insert) to insert the **AUTOEXEC OUTREAD** with a KEY/WHERE of **XFER-EMPL-ID**, the user exec **READ**, and the user exec **CREATE**.



```
UPDATE DATA GROUP ---- TRDISP.SD                              SIZE 000015 COL 01
COMMAND ===>                                                 SCROLL ===> CSR

         LABEL     REQUEST              KEY/WHERE                 IGNORE
====== ======== ============ ================================= ==============
VSAM=> TRGEMPLV
 REC=>                @DEFINE
         +
 0001 AUTOEXEC        OUTREAD         XFER-EMPL-ID
====== ======== ============ ================================= ==============
PCB==> TRGDBDV1       TRGDBDV1-PCB
 SEG=> TRGEMPL        @DEFINE         @XFER-EMPL-ID
 0001                 READ
 SEG=> TRGTASK        @DUMMY          @XFER-TASK-KEY
 SEG=> TRGTIME        @DUMMY          @XFER-TIME-KEY
====== ======== ============ ================================= ==============
DB2==> TRGEMPL1               TELON.TRGEMPL1
 ROW=> TRGEMPL1       @DEFINE         @EMPL-ID
 0001                 CREATE
 ROW=> XTRGEMPL       @DUMMY          @EMPL-ID
     TRGEMPL1
  ****** ******** ************ ********* BOTTOM OF DATA *********
     **************
```

After you press Enter, CA Telon redisplays the Update Data Group screen.



```
UPDATE DATA GROUP ---- TRDISP.SD                              SIZE 000015 COL 01
COMMAND ===>                                                 SCROLL ===> CSR
         LABEL     REQUEST              KEY/WHERE                 IGNORE
====== ======== ============ ================================= ==============
VSAM=> TRGEMPLV
 REC=>                @DEFINE
         +
 0001 AUTOEXEC        OUTREAD         XFER-EMPL-ID
====== ======== ============ ================================= ==============
PCB==> TRGDBDV1       TRGDBDV1-PCB
 SEG=> TRGEMPL        @DEFINE         @XFER-EMPL-ID
 0001                 READ
 SEG=> TRGTASK        @DUMMY          @XFER-TASK-KEY
 SEG=> TRGTIME        @DUMMY          @XFER-TIME-KEY
====== ======== ============ ================================= ==============
DB2==> TRGEMPL1               TELON.TRGEMPL1
 ROW=> TRGEMPL1       @DEFINE         @EMPL-ID
 0001                 CREATE
 ROW=> XTRGEMPL       @DUMMY          @EMPL-ID
     TRGEMPL1
  ****** ******** ************ ********* BOTTOM OF DATA *********
     **************
```

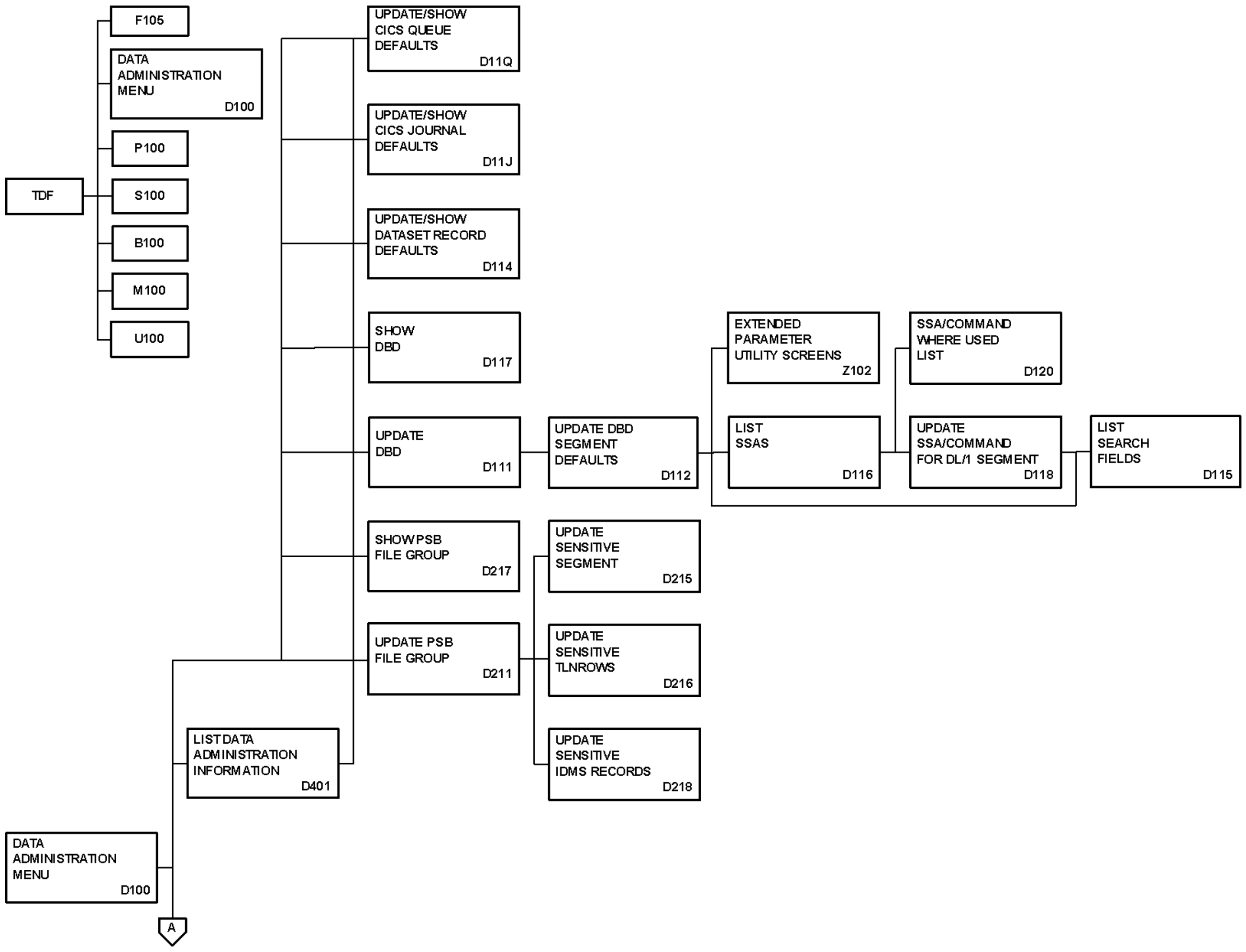
Press PF3 on the Update Data Group screen to complete the process. CA Telon returns you to the Update Screen Definition screen and displays the message **END PROCESSING PERFORMED** in the upper right-hand corner.

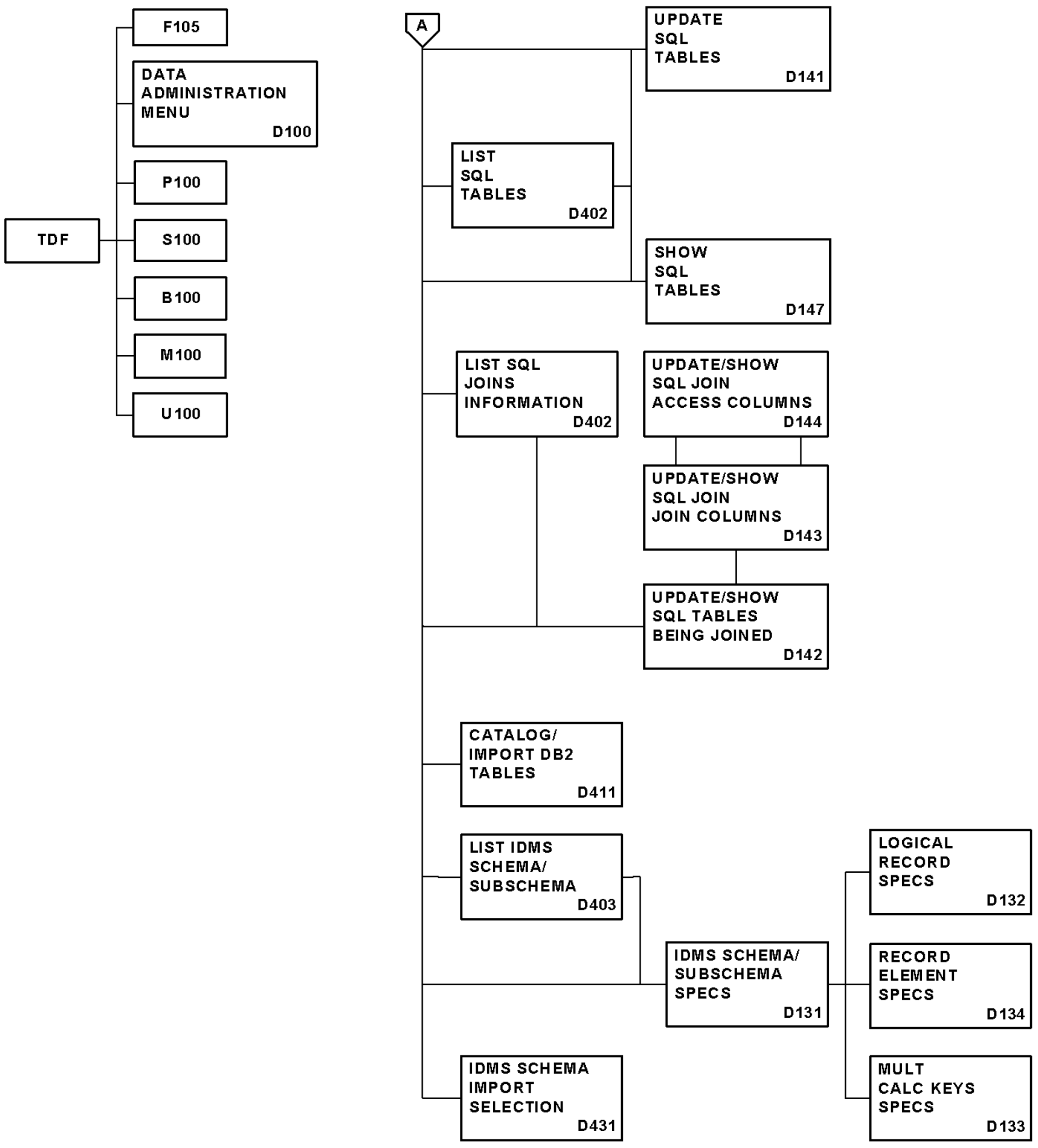
Press PF3 again to return to the Online Program Definition menu. CA Telon saves the screen definition and displays the message **SCREEN DEFINITION SAVED** in the upper right-hand corner.

Appendix A: Data Administration Screen Flow

The following diagram show the screen flow of the Data Administration environment.

Data Administration Screen Flow


F105
DATA ADMINISTRATION MENU D100
P100
TDF
S100
B100
M100
U100
UPDATE/SHOW CICS QUEUE DEFAULTS D11Q
UPDATE/SHOW CICS JOURNAL DEFAULTS D11J
UPDATE/SHOW DATASET RECORD DEFAULTS D114
SHOW DBD D117
UPDATE DBD D111
UPDATE DBD SEGMENT DEFAULTS D112
EXTENDED PARAMETER UTILITY SCREENS Z102
LIST SSAS D116
SSA/COMMAND WHERE USED LIST D120
UPDATE SSA/COMMAND FOR DL/1 SEGMENT D118
LIST SEARCH FIELDS D115
SHOW PSB FILE GROUP D217
UPDATE SENSITIVE SEGMENT D215
UPDATE PSB FILE GROUP D211
UPDATE SENSITIVE TLNROWS D216
LIST DATA ADMINISTRATION INFORMATION D401
UPDATE SENSITIVE IDMS RECORDS D218
DATA ADMINISTRATION MENU D100
A


TDF
F105
DATA ADMINISTRATION MENU D100
P100
S100
B100
M100
U100
A
UPDATE SQL TABLES D141
LIST SQL TABLES D402
SHOW SQL TABLES D147
LIST SQL JOINS INFORMATION D402
UPDATE/SHOW SQL JOIN ACCESS COLUMNS D144
UPDATE/SHOW SQL JOIN JOIN COLUMNS D143
UPDATE/SHOW SQL TABLES BEING JOINED D142
CATALOG/ IMPORT DB2 TABLES D411
LIST IDMS SCHEMA/ SUBSCHEMA D403
IDMS SCHEMA/ SUBSCHEMA SPECS D131
LOGICAL RECORD SPECS D132
RECORD ELEMENT SPECS D134
MULT CALC KEYS SPECS D133
IDMS SCHEMA IMPORT SELECTION D431

Index