

## Program Models

<b>Overview .....</b>	<b>253</b>
<b>Program Model Types.....</b>	<b>255</b>
<b>Creating Applications with Program Models .....</b>	<b>262</b>

## Overview

A *program model* is a predated, pretested, outline of program logic that can be used as a starting point for creating programs that execute under the control of Application Services. Models are written in either COBOL or C.

This chapter describes the various categories of ABC program models, and broadly defines each model. The process of using these models to develop applications is also discussed.

Detailed information for each program model, by category, is provided in the following chapters. When using these chapters, we recommended you concurrently view your program source code as these chapters are not meant to teach coding but be a reference for the functionality offered.

## Program Model Format

Program models are formatted to include shell code and application-specific code.

### Shell Code

Shell code contains the high-level organization for the model. For example, the shell code for any DEF program model provides the control structure to query the Dialog API and invoke an application's Display or Process side logic. The shell code also provides initialization and wrap-up logic common to the type of application and includes, where appropriate, opening and closing database transactions.

### Application-Specific Code

Application-specific code is placed in paragraph headers provided with the models. Application paragraphs are referenced in the shell code and cannot be removed or renamed without modifying the shell code. In some models, example code is included in the application paragraphs to demonstrate where to place functionality. This code indicates how the application interfaces with the shell code and the architecture. Study the example code before replacing it with code for your business application.

The format of program models is further defined by the coding language.

### ***COBOL Program Models***

The Data Division of each COBOL program model is formatted to contain:

- Working Storage comprised of copybooks, generic variables, and literal definitions referenced in the Procedure Division.
- Linkage comprised of copybooks that provide the interface to Dialog Services, Presentation Services, and the Error Handler.

The Procedure Division of each COBOL program model is formatted to contain:

- References to shell code paragraphs containing the high-level organization for the model. For example, shell code often performs initialization and wrap-up functions. Do not modify these paragraphs as they contain pretested logic central to the execution of the model.
- Application code paragraphs where you can modify example code and place application-specific code. These paragraphs are referenced in the model by name and paragraph number. Example code shows how the application code interfaces with shell code and Application Services. Use the example code as a reference before replacing it with application-specific code.

### ***C Models***

Program models written in C are similar in structure to the COBOL models. All include statements referenced in the application module, and included to interface with Application Services, are provided. Sections of C code that relate to COBOL paragraphs perform the same functions as their COBOL counterparts, and are referenced by the same paragraph numbers in the comments and error messages.

## **Program Model Benefits**

Using program models minimizes coding time and ensures program maintainability. Specifically, these models:

- Create uniformity by offering you a common program structure.
- Reduce application programming effort by providing example code.
- Decrease the amount of testing because all shell code is free of errors.
- Provide code that is portable to other platforms with a minimum of change.

## Program Model Types

Choose program models based on your programming language and the model functionality. Each program model is classified according to the following categories:

- ACMS
- Conversation Support
- Embedded SQL
- Portable
- C

The following sections describe each of these categories.

### ACMS Program Models

ACMS program models, written in COBOL, allow you to create ACMS modules that perform specific tasks when a server starts, encounters a fatal error, or ends. These modules are in addition to the conversation modules that can be defined using the DEF facility. Also, these modules do not have any windows associated with them and are not part of any conversation.

The table below describes the ACMS program models.

Model	Filename	Description
ACMS Initialization Procedure	AABC.MDL	Gives your application the ability to open files and prepare the database.
ACMS Task Cancel/Abend	ADEF.MDL	Ensures your application performs all error-handling, clean-up processing, and sign-off for user-specified fatal errors and cancellations.
ACMS Termination/Cancel	AGHLM.DL	Gives your application the ability to close files and release data.

Refer to "[ACMS Program Models](#)" for more information.

## Conversation Support Program Models

The Conversation Support program models, written in COBOL, allow you to create modules that provide support functions for conversations. These functions are not normally handled within the step of a conversation.

The table below describes the Conversation Support program models.

Model	Filename	Description
CCP Server Front End	CCPABC.MDL	<p>Contains the control logic that ensures your application server:</p> <ul style="list-style-type: none"> <li>Registers with Distribution Services.</li> <li>Sends and receives messages to and from Dialog Services.</li> <li>Exits in an orderly fashion.</li> </ul> <p>If you call any C step program from within a server front end, you must use the C Server Front End model and not this model.</p>
Generic Server Front End	CCPDEF.MDL	<p>Contains the same control logic as the CCP Server Front End model but does not assume the data being processed is from the Conversation Control Record (CCR). Use this model for asynchronous processing or other processing that does not use the Dialog API.</p>
List Box Callback	CCPHIJ.MDL	<p>Provides logic so you can create list box select programs for your application fields. This module is valid for both JKL and MNO applications and can access either Rdb tables or RMS files.</p>
RMS Begin/End/Cancel	CCPJKL.MDL	<p>Allows your application to perform housekeeping or cleanup functions for conversations. When a conversation is accessed or conversation flow branches because of a Cancel, End, or Fatal Error condition, this module calls the appropriate beginning-of-conversation, end-of-conversation, or cancel routine.</p>

Refer to “[Conversation Support Program Models](#)” for more information.

## Embedded SQL Program Models

Embedded SQL program models, written in COBOL, allow you to create step modules that access data stored in Rdb tables. Specifically, you can create detail, list, prompt, and select modules.

These program models are specific to the ABC environment. Do not use them to get information from RMS files or if you want your application to be portable to databases other than Rdb. Example SQL is included in these models to help you code the proper syntax.

The table below describes the Embedded SQL program models.

Model	Filename	Description
SQL Detail	SQLABC.MDL	The basic online model for most application windows. Allows your application to perform complete data validation and posting. This module is not called until all key information is entered and verified. Example SQL calls are provided for transferring information to and from the database.
SQL Prompt	SQLDEF.MDL	Usually the first step of an application conversation. Allows your application to accept data entered from a terminal for processing. Entered data passes to a Select, List, or Detail module. Example SQL calls are provided for transferring information to and from the database.
SQL Select	SQLGHI.MDL	Provides logic so your application can scroll backward or forward through a file to search for a complete or partial key. Includes code permitting you to perform action on selected data. Example SQL calls are provided for transferring information to and from the database.

Refer to [“Embedded SQL Program Models”](#) for more information.

## Portable Program Models

Portable program models, written in COBOL, allow you to create step modules that access data stored in either Rdb tables or flat files. Specifically, you can create detail, list, prompt, and select modules.

These program models are as generic as possible to allow movement to other platforms with a minimum of conversion. If you are creating portable applications, it is recommended you read the *Portability Guide* for additional coding considerations.

The table below describes the Portable program models.

Model	Filename	Description
Portable Detail	PORTABC.MDL	The basic online model for most application windows. Allows you to build portable applications that perform complete data validation and then post valid data to the database. This module is not called until all key information is entered and verified.
Portable List	PORTDEF.MDL	Provides logic so you can build portable online applications that scroll backward or forward through a file to search for a complete or partial key. Selected data is read only and cannot be updated.
Portable Prompt	PORTGHI.MDL	Usually the first step of an application conversation. Allows you to build portable online applications that accept data entered from a terminal for processing
Portable Select	PORTJKL.MDL	Provides logic so you can build portable applications that scroll backward or forward through a file to search for a complete or partial key. Includes code permitting you to perform action on selected data.

Refer to [“Portable Program Models”](#) for more information.

## C Program Models

C program models allow you to create modules written in C. You can create detail and prompt step modules that access data stored in either relational database tables or native flat files. You can also create server front end modules. C program models are as generic as possible to allow movement to other platforms with a minimum of conversion.

The table below describes the C program models.

Model	Filename	Description
Detail	CABC.MDL	The basic online model for most application windows. Allows your application to perform complete data validation and then post valid data to the database. This module is not called until all key information is entered and verified.
Prompt	CDEF.MDL	Usually the first step of an application conversation. Allows your application to accept data entered from a terminal for processing. Entered data passes the Detail module.
Server Front End	CGHI.MDL	Contains control logic that ensures your application server: <ul style="list-style-type: none"> <li>Registers with DEF Services.</li> <li>Sends and receives messages to and from GHI Services.</li> <li>Exits in an orderly fashion.</li> </ul> <p>If you call any C step program from within a server front end, you must use this model and not the COBOL CCP Server Front End model.</p>

Refer to "[C Program Models](#)" for more information.