SCLM Administration Just Got Easier
April 2009

A Practical Guide to the SCLM Administrator Toolkit

SCLM Administrator Toolkit v5.1

Jennifer Nelson
Product Specialist
Rocket Software, Inc.

CONTENTS

Contents ii

List of Figures iv

List of Examples vii

List of Tables viii

Revision History 1

1 What is SCLM? 1

2 Why use the SCLM Administrator Toolkit? 1

3 A Practical Case Study 3

3.1 A Brief Introduction 3

3.2 Create a new project 6

3.3 Define the Types 12

3.4 Define the Groups 14

3.5 Select the Group/Type Data Sets 17

3.6 Add Language Definitions 19

3.7 Customize the DB2 CLIST 27

3.8 Build the Project 29

3.9 Create a Project Filter 31

3.10 Migrate Project Assets 34

3.11 Create Architecture Definition 39

3.12 Test the project 54

3.13 Deploy the project 63

4 A Case for the Administrator Toolkit 67

Appendix A: Create the objects 69

Appendix B: The COBOL program 72

LIST OF FIGURES

Figure 1: Nodes within SCLM Administrator Toolkit. 4
Figure 2: Create a new project. 6
Figure 3: Provide a project name and sandbox name. 7
Figure 4: Specify the project definition data sets to use to create the project. 8
Figure 5: Specify all data sets that contain SCLM macros required to build the project. 9
Figure 6: Review the information that will be used to create your project. 10
Figure 7: The status of each task is displayed for your information. 11
Figure 8: The Project Editor displays the newly created project, and editing can now begin. 11
Figure 9: Multiple Types can be selected by pressing the CTRL key. 12
Figure 10: Define each data sets' parameters. 13
Figure 11: A sample SCLM project hierarchy. 15
Figure 12: Specify the groups to include in your project's hierarchy. 15
Figure 13: The completed project hierarchy showing the direction of promotion. 16
Figure 14: Tabular form displays each group's parameters in an easy-to-view list. 17
Figure 15: Sample hierarchy illustrating different groups requiring different kinds of data sets. 18
Figure 16: Select the group/type combination data sets to include in your project. 18
Figure 17: Select the option you wish to use to create a Language Definition. 20
Figure 18: A message is returned when a language is successfully added to a project. 21
Figure 19: Specify a language to copy, but edit the language to tailor it to your project. 21
Figure 20: Expand the data set name and select the member name to use. 22
Figure 21: Make any necessary edits to the language definition to tailor it to your project. 23
Figure 22: The Translator Summary provides a list of translators defined within the language. 24
Figure 23: All FLMTRNSL macro parameters can be edited. 25
Figure 24: Edit the data sets concatenated within the FLMCPYLB macro. 26
Figure 25: Edit the Copy data set name as necessary. 26
Figure 26: The list of language definitions is displayed within a list. 27
Figure 27: Automatically create the DB2 CLIST which drives the plan or package BIND process. 28
Figure 28: Provide BIND syntax parameters for the DB2 applications within your project. 28
Figure 29: Select the option to Build your project. 29
Figure 30: Click OK to build your project's definition into a load module. 30
Figure 31: The status indicator provides a progress report of the current task. 30
Figure 32: Messages are displayed within a pop-up window with additional information. 31
Figure 33: Create a project filter to view your newly created project. 31

Figure 34: Provide the project filter criteria. 32
Figure 35: Refresh the node to view the new project filter. 32
Figure 36: Expand the new project filter to see all projects that match the filter name. 33
Figure 37: Right click your project to navigate to the Migration Wizard. 34
Figure 38: Click the button Add to add an asset to the list. 35
Figure 39: Specify the data set that contains the assets you want to migrate. 36
Figure 40: Select the data set that contains the assets to migrate. 36
Figure 41: Select the members you want to include in the list. 37
Figure 42: The selected assets to migrate are listed within the Migration Wizard. 37
Figure 43: Specify the options to use to migrate the assets. 38
Figure 44: The Migration Report is returned for your review. 39
Figure 45: Invoke the Architecture Definition Wizard. 40
Figure 46: Select which kind of Architecture Definition to create. 41
Figure 47: Specify the name and data set type for each kind of architecture definition. 42
Figure 48: Create the Compilation Control architecture definition. 44
Figure 49: Create the Link Edit Control architecture definition. 45
Figure 50: Create the DB2 BIND architecture definition. 46
Figure 51: Create the High Level architecture definition. 47
Figure 52: The compilation control architecture definition is returned for your review. 48
Figure 53: The link edit control architecture definition is returned for your review. 48
Figure 54: The DB2 bind architecture definition is returned for your review. 49
Figure 55: The DB2 CLIST executes the binds when the DB2 Bind Architecture Definition is built in SCLM 50
Figure 56: The High Level architecture definition is returned for your review. 51
Figure 57: Specify the parameters with which to create the architecture definitions. 52
Figure 58: Informational messages are returned about the create process. 52
Figure 59: View or edit architecture definitions. 54
Figure 60: Specify the SCLM project whose assets you want to access. 55
Figure 61: Select option 1 to view the contents of a library. 56
Figure 62: Specify the data set type where the architecture definition is stored. 56
Figure 63: Build the architecture definition by selecting option C. 57
Figure 64: Double check the asset specified is the one you want to build. 57
Figure 65: Promote the architecture definition by selecting option P. 59
Figure 66: Double check the asset specified is the one you want to promote. 60
Figure 67: Select to deploy the sandbox project. 64
Figure 68: Export the new project's definition. 64
Figure 69: Informational messages are returned about the deployment. 65

Figure 70: Create a new project filter. 65
Figure 71: Provide the criteria for the project filter. 66
Figure 72: Refresh the Deployed Projects node to see the new project filter. 66
Figure 73: Collapse, then re-expand the node Deployed Projects to see the new project filter. 67

LIST OF EXAMPLES

Example 1: SCLM Build report from high level architecture definition AUZDBSEL. 58
Example 2: SCLM Promote report from high level architecture definition AUZDBSEL. 60
Example 3: Results of executing sample program AUZDBSEL from the new project. 63
Example 4: Project DDL to create the necessary DB2 objects for this project. 69
Example 5: Project DML for the first table to insert the necessary DB2 data for this project. 69
Example 6: Project DML for the second table to insert the necessary DB2 data for this project. ... 70
Example 7: Sample JCL to create the DB2 objects and populate the data. 70
Example 8: Sample COBOL program for the DB2 project. 72

LIST OF TABLES

Table 1: The features and functions of the Administrator Toolkit and the benefits they provide..... 2

1 What is SCLM?

One of the biggest investments of any company is the application source code that is critical to conducting everyday business. Application source code can be easily maintained and secured using IBM's Software Configuration and Library Manager, which is packaged as an integrated component of ISPF. SCLM is a comprehensive tool that has two major functions: configuration management and library management.

As a configuration manager, SCLM knows how all of the different pieces of an application's source code fit together. Not only can source, object and load modules be managed by SCLM, but any asset that resides within the application's set of libraries can be managed using SCLM, such as documentation and JCL.

Using parameters defined to SCLM, it can translate inputs into outputs, and can drive almost any procedure that translates an input to an output. Source modules can be 'linked' or 'related' to other modules such that when a change is made to one module, SCLM ensures the 'related' modules are all translated after a change is made. SCLM can also quantify the impact of a change to an application's source code.

As a library manager, SCLM controls the development cycle of the application by ensuring translated modules are promoted to the next level within the development hierarchy. In addition, SCLM can audit and track changes that are made over time to each module within the project, allowing you to fall back to a previous version of a module if an undesired change was made. SCLM can lock a module while under development to prevent concurrent work on the same piece of source code.

2 Why use the SCLM Administrator Toolkit?

Defining an application's source code within SCLM to create a project requires that the user understand the SCLM parameters and their valid options and combinations to create the desired behavior when translating an application's source code modules into executable programs. SCLM provides no user interface for creating or editing a project's definition, leaving the user to manually provide the appropriate macro parameters and their options within the proper format.

To an inexperienced SCLM project administrator, editing an SCLM project can become a daunting task, as one small change to a translator can result in project errors when used with SCLM. In addition, an experienced SCLM project administrator might prefer a more efficient and user-friendly interface, eliminating manual errors and reducing the amount of time it takes to edit an existing project's definition, or create a new project definition. Every SCLM macro, parameter and corresponding value are supported from within the Administrator Toolkit, allowing users to create simple or even complex projects.

The Administrator Toolkit is a product meant to ease the administration process by providing a rich task-oriented user interface in either traditional ISPF panels or a graphical user interface. Both new and experienced SCLM project administrators alike can benefit from content-sensitive wizards that help prevent conflicting parameter values from being defined within a project. In

addition, descriptive panel Help explains the purpose of each wizard and explains the details of the parameters.

The following table lists the features and functions of the Administrator Toolkit and their benefit to SCLM administrators in maintaining their SCLM project definitions.

Table 1: The features and functions of the Administrator Toolkit and the benefits they provide.

Feature	Function	Benefit
Project Editor	User-friendly guided interface to create new projects, or edit existing projects.	Users new to SCLM can be productive without understanding the logic behind SCLM's features and functions.
Projects in a Sandbox	Modify the project's definition as much or as little as necessary without impacting production projects.	Define and test a project's definition in a safe environment without impacting production. Ensure that users who follow the rules only deploy working projects into production.
Architecture Definition Wizard	Create architecture definitions from JCL, from load modules, or from the guided wizard.	Users are not required to know the nuances of how to create an architecture definition. Instead, the Architecture Definition Wizard can create an architecture definition from a variety of sources. Or, if an experienced user would prefer, they may create the architecture definition from the wizard which guides them through the process.
Language Definition Wizard	Create language definitions from existing JCL, from load modules, from existing language definitions, or create them from within the guided wizard.	Language definitions can be hard to get right. Using the content-sensitive fields within the Language Definition Wizard, languages are created in less time with fewer errors.
Clone an existing project	Copy an existing project's definition and optionally the project	Projects can contain thousands of assets. With one keystroke or the click of one button, the Administrator Toolkit can automatically copy everything within a project for inclusion in another.
Migration Wizard	Migrate assets into a project using the guided wizard interface.	Users can pick and choose which assets to include from another project, or outside of SCLM control using the guidance of the task-oriented user interface. The Administrator Toolkit effortlessly copies assets into the SCLM project and registers the assets with the SCLM accounting dataset.
Remote Migration Wizard	Migrate remote assets into a project using the guided wizard interface.	Users can easily migrate assets that reside on their PC or another remote location and maintain the assets within SCLM with no extra effort.
Administer EAC Profiles	Track EAC violations, create new EAC applications and profiles, edit existing ones, or delete them.	The Administrator Toolkit provides users an intuitive graphical user interface to EAC to administer EAC profiles within the same GUI interface as they administer their SCLM projects.
Administer RACF profiles	Create new RACF Data Set Profiles, Edit, and Delete existing RACF Data set Profiles.	The Administrator Toolkit provides an intuitive graphical user interface to manage RACF data set profiles.

Feature	Function	Benefit
Maintain a project's VSAM datasets	Create, maintain, and run arbitrary REXX Execs that perform essential functions on SCLM-managed libraries. A list of REXX Execs is provided to get the administrator started performing common tasks.	SCLM project administrators can either execute sample REXX Execs, or add their own scripts to the menu, providing flexible access to listing and running those tasks most often used.
Build a project definition	Automatically assembles project definitions with one click.	Assemble and link JCL doesn't need to be maintained. Rather, the Administrator Toolkit automatically creates all the components required by SCLM to recognize the project as an SCLM project.
Project Explorer	Automatically discover existing SCLM Projects to administer.	Users can find the project they are interested in administering from a list without having to know its location in the system.

3 A Practical Case Study

This white paper will focus on creating a new DB2 project definition from scratch using the Project Editor. Existing language definitions will be copied into the project using the Language Definition Wizard, but will be edited for the needs of this specific case. Assets from another project will be migrated into this project using the Migration Wizard. The appropriate architecture definitions required for a DB2 project will be created using the Architecture Definition Wizard.

The new project will be built by the Project Editor and then used from within SCLM to demonstrate the relationship between the SCLM Administrator Toolkit and SCLM.

The graphical user interface will be used for the purpose of this white paper. This white paper makes the assumption that the Eclipsed-based graphical user interface has already been installed and configured, as no further configuration instructions will be provided. For detailed instructions on installing and configuring the graphical user interface, please refer to the *IBM SCLM Administrator Toolkit Installation and Configuration Guide, SC23-9567-00*, in the chapter entitled "Install and configure the Eclipse workstation client."

3.1 A Brief Introduction

After logging into the host connection on which to create a new SCLM project, the Remote System View will list those SCLM projects that already exist. Figure 1 displays the nodes that display beneath SCLM within the Remote Systems View.

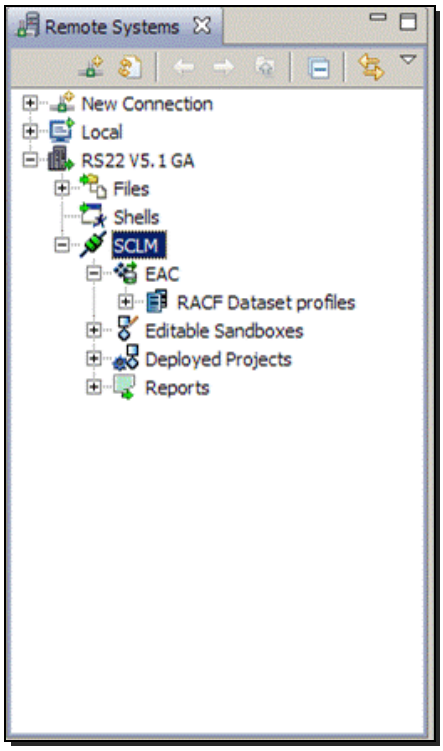

Figure 1: Nodes within SCLM Administrator Toolkit.

When a host connection is made, the icon next to SCLM will appear green, and will be connected. When a host connection is disconnected, that is, when you log off of a host connection, the icon will appear red and separated.

The EAC icon allows you to access IBM Enhanced Access Control for SCLM, if it has been installed on the host. However, you can still use the Administrator Toolkit's RACF Dataset Profile functions without having Enhanced Access Control installed.

Within the EAC node, EAC profiles, application/function pairs, and high/low program pairs can be created, edited, or deleted. Likewise, EAC violations can be tracked and viewed from within the EAC node. For more information on the administration and management of EAC profiles and applications, refer to the *IBM SCLM Administrator Toolkit User's Guide*, *SC23-9566-00*, in the chapter entitled "Administer EAC and RACF".

When the EAC node is expanded, the RACF Dataset Profile node is displayed as shown in Figure 1 on page 4. Using the interface the Administrator Toolkit provides, you can view, edit and create RACF Dataset Profiles. However, you must have RACF Special authority to use edit and create new RACF dataset profiles. For more information on the administration and management of RACF Dataset profiles and applications, refer to the *IBM SCLM Administrator Toolkit User's Guide*, *SC23-9566-00*, in the chapter entitled "Administer EAC and RACF". For more information on the RACF profile options available, refer to the *IBM Security Server RACF General User's Guide*.

The icon titled Editable Sandboxes, when expanded, will display a list of SCLM projects whose definitions are in test and are not in production. Projects listed in Editable Sandboxes are in a separate contained environment for the purpose of testing the behavior of the project without affecting production. When a project's test results are successful, it can be deployed into production. When a project is deployed, it will no longer appear in the list of Editable Sandboxes; rather, it will appear in the list of Deployed Projects when the Deployed Project list

is refreshed. Alternate projects are fully supported with sandboxes. All SCLM project parameters are available and can be used to define alternate projects within the Administrator Toolkit.

The icon titled Deployed Projects, when expanded, lists those SCLM projects whose definitions are being actively used in production. The associated sandbox for these projects no longer exists, and therefore cannot be tested without affecting production. If a project must be edited to change its definition, you may either create a sandbox for your project in which to test your edits, or make edits directly into the deployed project. When you edit a deployed project, it will appear in the list of Editable Sandboxes for the duration of time the project is being edited. Once your edits are complete, you must re-deploy your project for the edits to take effect. Deployed alternate projects are treated in the same manner as full projects, in that deployed alternate projects can be edited at any time and re-deployed when changes have been completed.

The last icon listed within the Remote Systems view is Reports. When expanded, this node will contain the results of the tasks you performed within the Administrator Toolkit. The reports are for your information and to assist customer support. To delete a report, simply expand the node to list all reports and highlight the one you want to delete. Right click the report, then select the option titled Delete. To keep the report even after you close your Administrator Toolkit GUI application, right click the report name, then select the option titled Make Persistent.

3.2 Create a new project

Creating a new project can be done in one of three ways. An existing project can be imported, cloned, or a new project can be created from the beginning. Importing an existing project is an operation you do if you have SCLM projects created without use of the Administrator Toolkit and want to manage the project with the Administrator Toolkit. Cloning an existing project allows you to copy not only the project's definition, but you may optionally choose to copy its corresponding group/type data sets and their contents as well as accounting, audit, and version history.

In this case, a new project will be created from scratch. Highlight the icon Editable Sandboxes, then right click to get an additional menu. Select **New** then select **SCLM Project**, as shown in Figure 2 below on page 6.

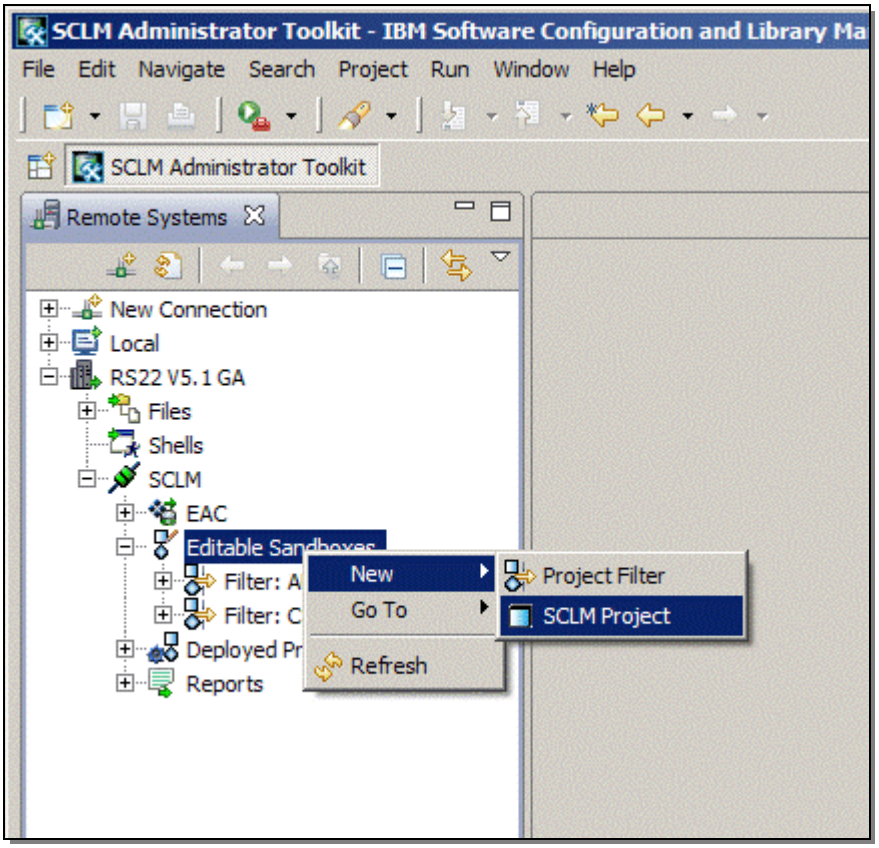

Figure 2: Create a new project.

A window titled Project Definition Wizard will display, allowing you to provide the project name. Because SCLM uses the project name as the high level qualifier of all project-related data sets, the project name you supply must follow MVS data set naming standards, as well as follow RACF rules within your shop. Also within the Project Definition Wizard window is a checkbox titled Create Sandbox, as shown in Figure 3 on page 7. It is this option that allows you to create your project within a test environment before deploying the completed project into production.

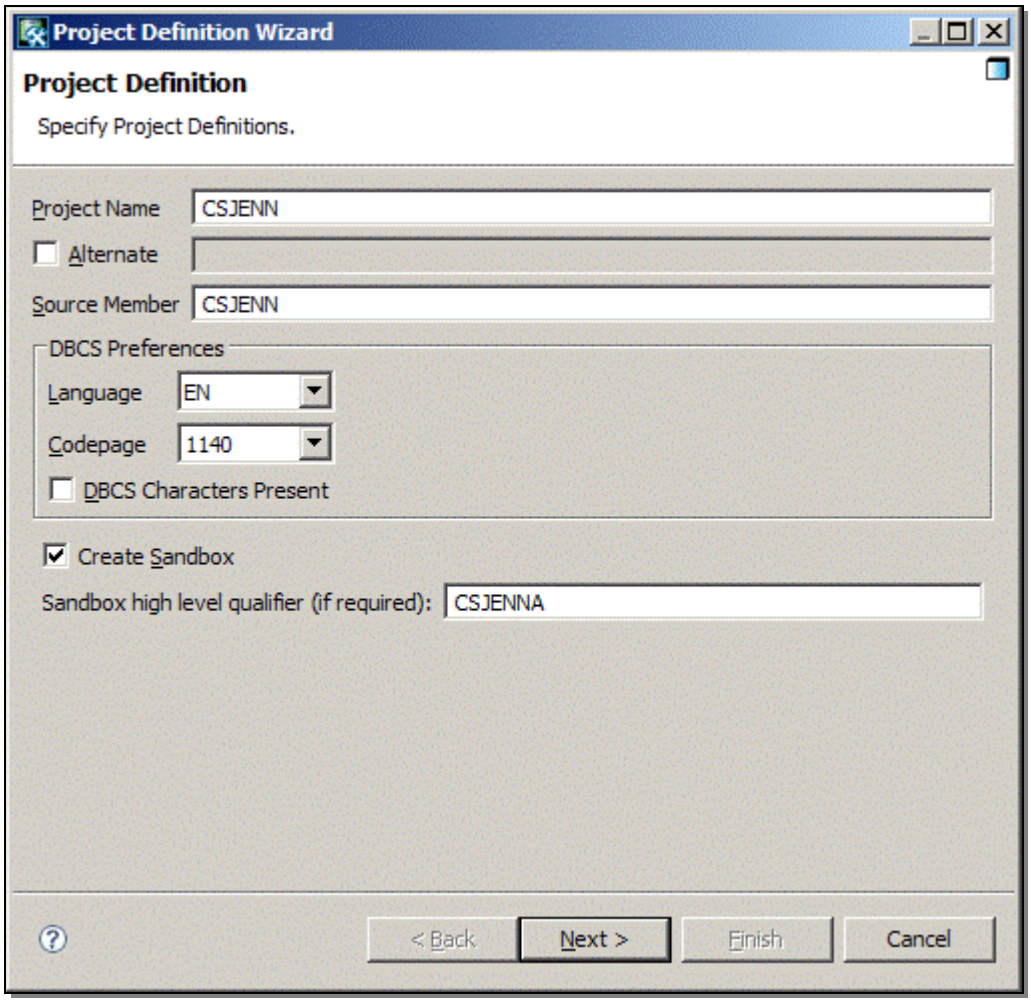

Figure 3: Provide a project name and sandbox name.

Provide a project name as well as a sandbox name. Both values will become the high level qualifier of the SCLM-managed data sets and should adhere to MVS data set naming standards as well as adhere to any RACF rules within your shop. That is to say, you must have RACF authority to create data sets that begin with the project name and sandbox name you provide. Click **Next** to continue.

Note: It is not recommended that you use a Codepage other than 1140 for your project definition.

The next window, titled Project Definition Data Sets, is displayed, allowing you to define the data sets that will be used in creating the project, as shown in Figure 4. These data sets are referred to as PROJDEFS data sets, and do not contain application source code; rather, these data sets will be used by the Administrator Toolkit to store the parameters and options specified to define the project, as well as the listings when the project's load module is assembled and linked.

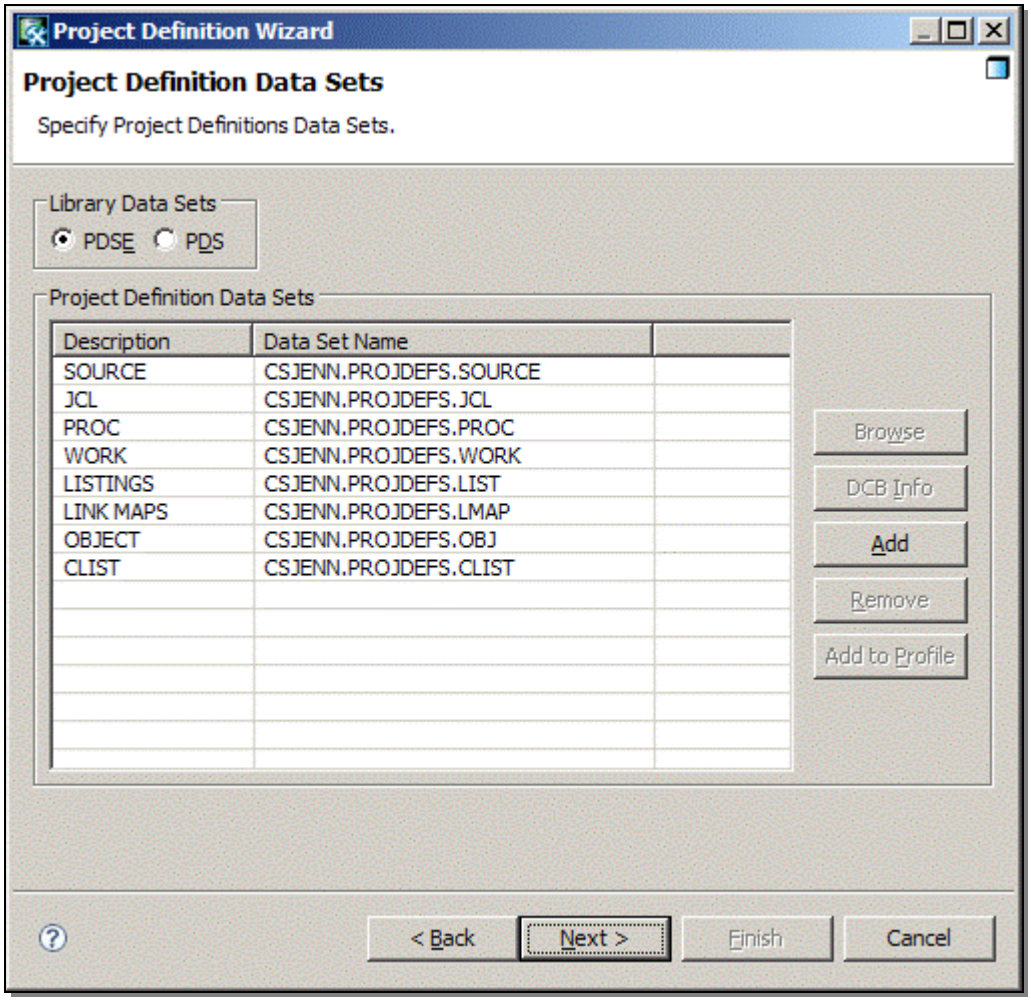

Figure 4: Specify the project definition data sets to use to create the project.

To change any of the data sets within this window, simply click on the data set name to activate the field. It is advised, however, that the data set high level qualifier not be changed, as it corresponds to the project's name. To add a data set to the list, you may click the button titled Add. In this case, the project's default PROJDEFS data sets will be used. Click **Next** to continue defining the project.

Note: There are several PROJDEFS data sets whose names cannot be changed since they are required for the correct operation of SCLM and the SCLM Administrator Toolkit. These are not shown on this screen and are the PROJDEFS.LOAD data set and PROJDEFS.SETTINGS.* set of data sets.

The next window, titled SYSLIB Data Sets, is displayed and lists those data sets that contain copybooks, language definitions, or other SCLM macros that must be resolved when assembling and linking the project's load module, as displayed in Figure 5.

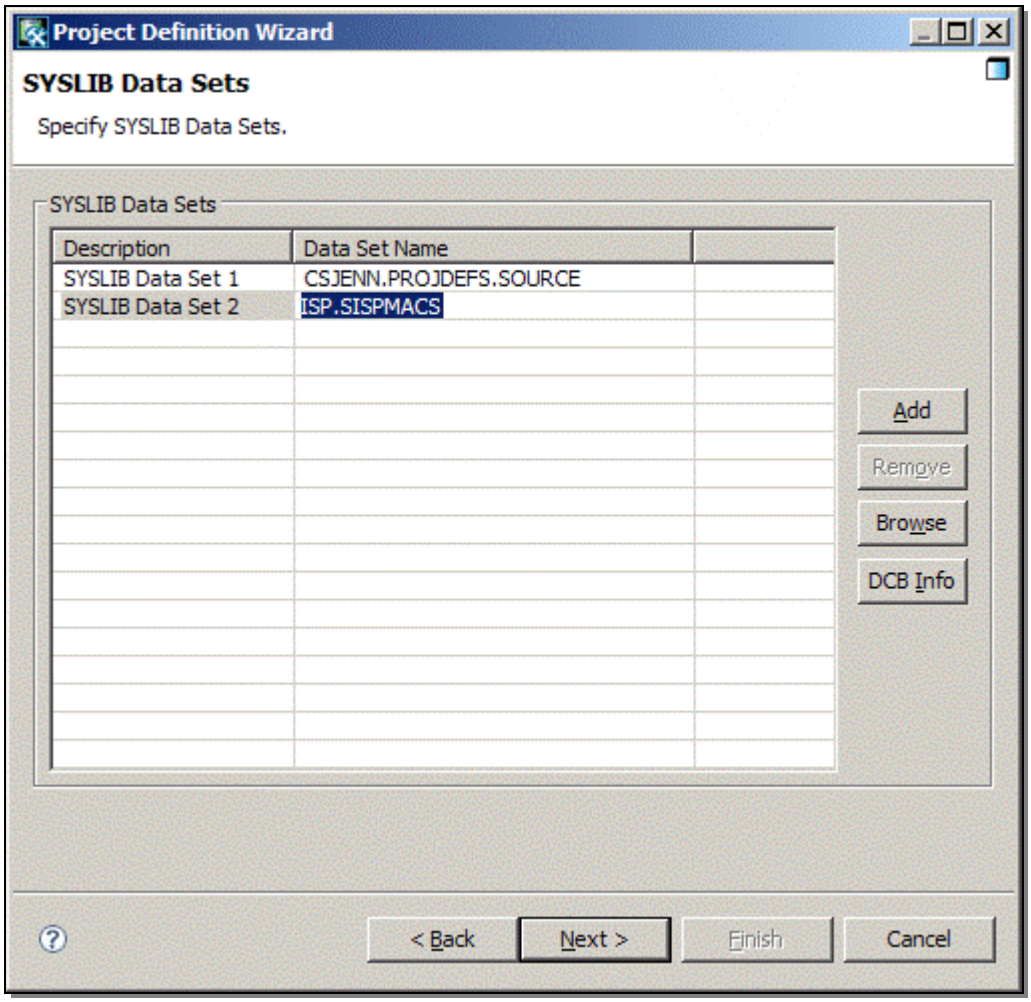

Figure 5: Specify all data sets that contain SCLM macros required to build the project.

The data set listed as SYSLIB Data Set 1 must be the project's SOURCE data set and cannot be changed. The data set listed as SYSLIB Data Set 2 is normally the SCLM macro library where most language definitions are stored. The data set name may be edited by placing your cursor on the data set name. This activates the field and allows you to edit the data set name. If there are additional copybooks or languages that reside in other data sets that you would like to include within the project definition to resolve dependencies, you may do so by clicking the button titled Add. Click **Next** when you have completed adding data sets to the SYSLIB concatenation.

The last Project Wizard panel, titled Project Definition Information, is displayed allowing you to review the information you provided to create the data sets associated with your project's definition, as shown in Figure 6.

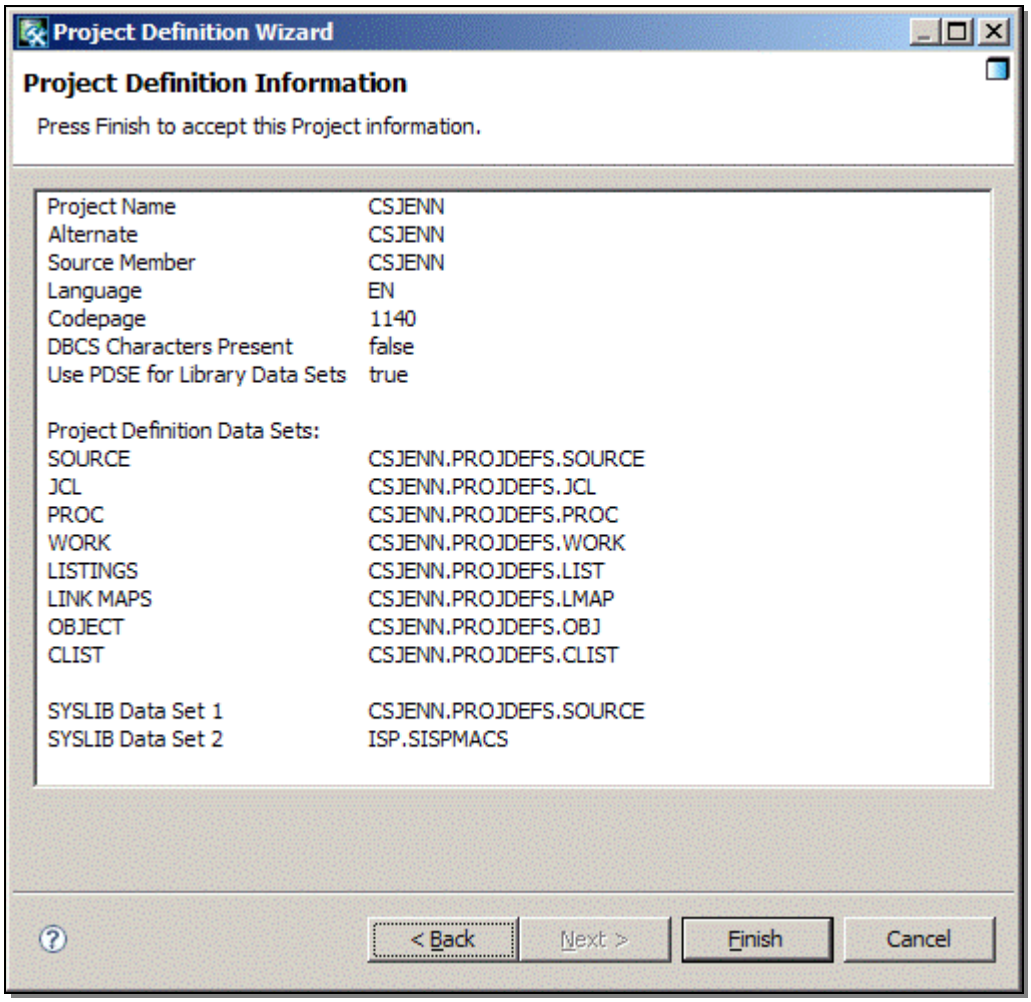

Figure 6: Review the information that will be used to create your project.

To make any changes to the data sets that will be used to create your project's definition, click the button titled Back. To complete your project's definition, click the button titled **Finish**. The Administrator Toolkit will begin creating the specified data sets, and will display the status of the task on the bottom right-hand side of the Administrator Toolkit perspective, as shown in Figure 7 on page 11.

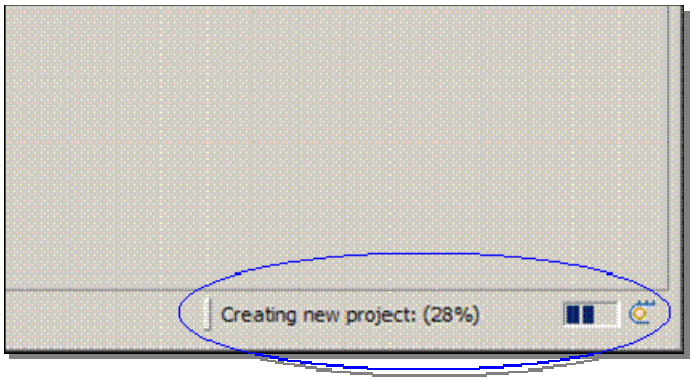

Figure 7: The status of each task is displayed for your information.

When the Administrator Toolkit has completed creating the project's definition data sets, the Project Editor window will display the new project, as shown in Figure 8.

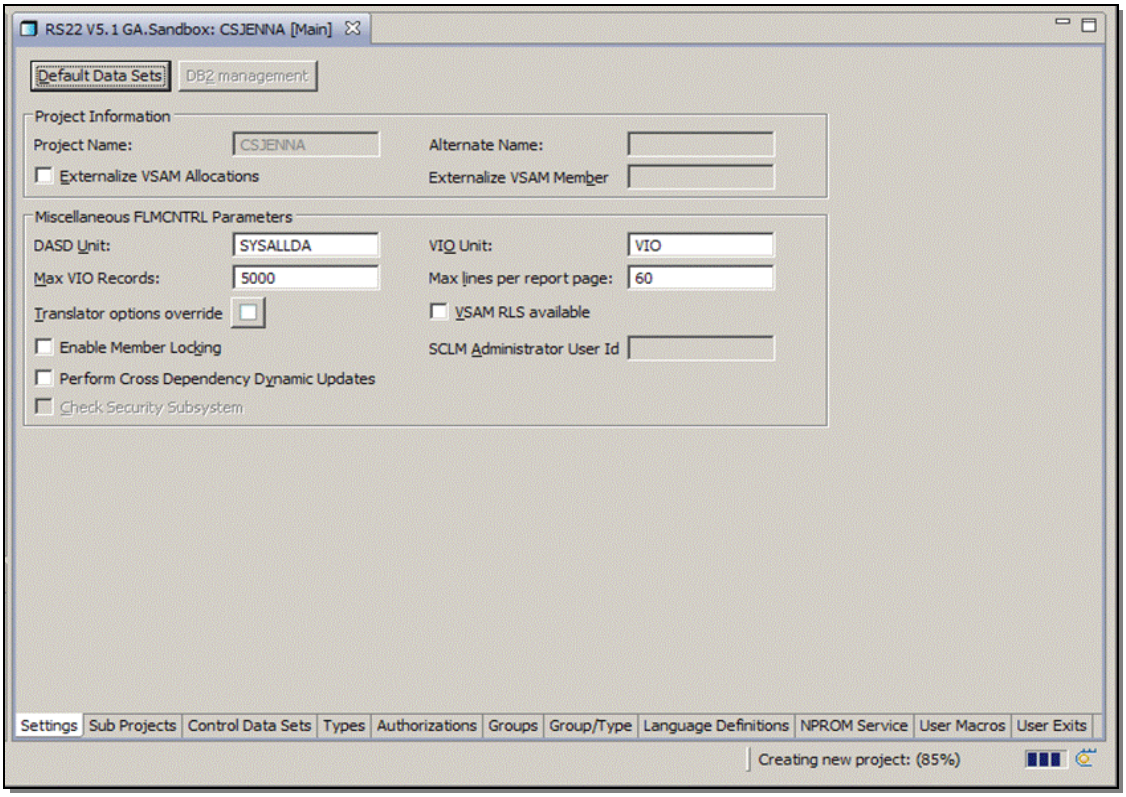

Figure 8: The Project Editor displays the newly created project, and editing can now begin.

Within the Project Editor, all SCLM macros and their corresponding parameters can be implicitly edited, added or deleted. The tabs across the bottom of the Project Editor allow you to jump from function to function as necessary to define your project. For example, the SCLM macro FLMLANGL pertains to language definitions. Therefore, clicking on the tab titled Language Definitions will allow you to edit parameters related to the FLMLANGL macro.

In this example, data set types will be added next, so click on the tab titled **Types** in the Project Editor.

3.3 Define the Types

Data set types are those kinds of data sets that you add to a project to store application modules of that type. For example, COBOL source code should be stored in a data set type called COBOL, while java source code should be stored in a data set type called JAVA. Each project will require different kinds of data sets to store different kinds of objects that are pertinent to the project being created. Planning ahead of time for your project will reduce the effort and time required to create and test your project.

On the tab titled Types, there are two frames; the left-hand frame displays those data set types that are pre-defined. That is, their data set parameters have already been specified, and they can be included in your project. The right-hand frame will list those data set types that are included in your project. Upon creating a new project, this frame will be empty, as shown in Figure 9.

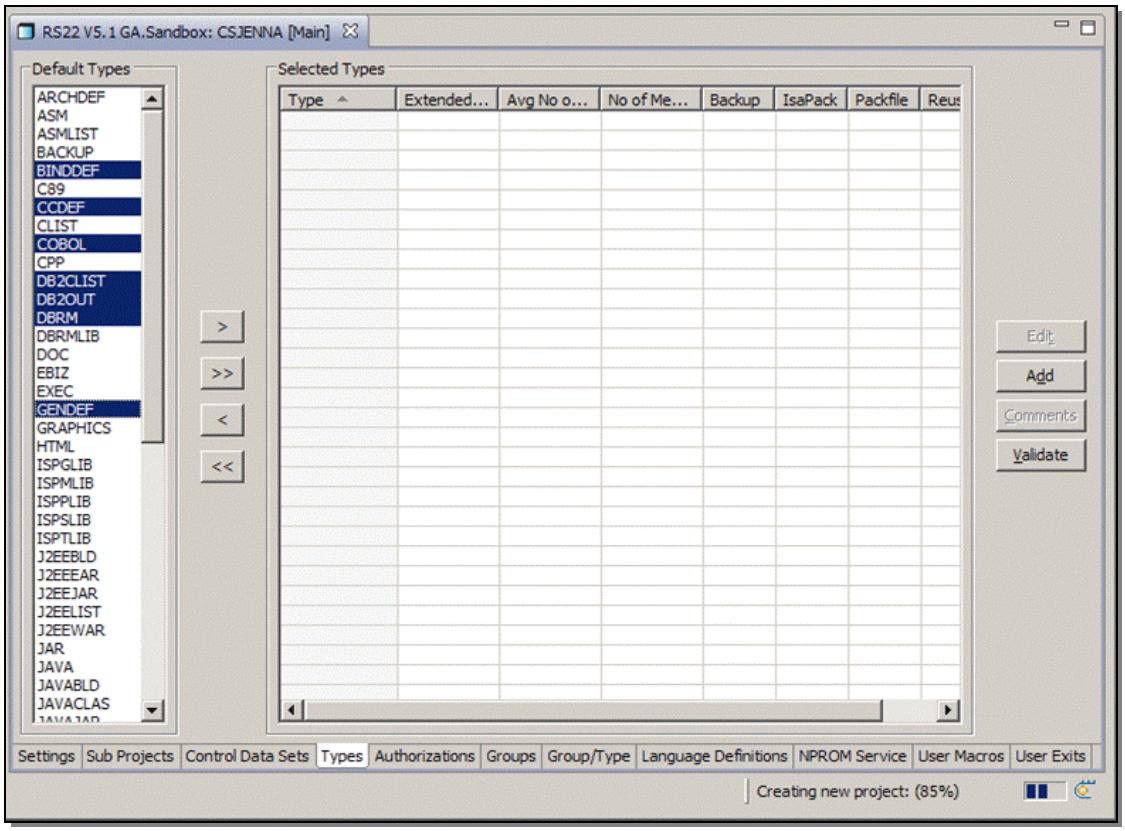

Figure 9: Multiple Types can be selected by pressing the CTRL key.

You may select many different data set types from the left-hand frame by pressing and holding down the CTRL key while you select the data set types with the mouse, then clicking the single right arrow to move the selected types into the right-hand frame. Or, you may simply click on one data set at a time, and select the single right arrow to move the selected data set to the right-hand frame. Clicking on the double right arrow will move all default data set types from the left-hand frame into the right-hand frame for inclusion into your project.

To include a data set type that is not listed within the Default Types frame, click the button titled Add on the right-hand side of the Types tab to add and define your own data set type. It will then appear in the list of selected types and will be included in your project.

In this example, we have included the data set types BINDDEF, CCDEF, COBOL, DBRM, DB2CLIST, DB2OUT, GENDEF, LECDEF, LIST, LKEDLIST, LOAD and OBJ.

When you have completed selecting the types of data sets to include in your project, you must then provide values for each of the data sets under the columns titled Avg No of Recs and No of Members to size each data set accordingly. The value you provide for No of Members is the number of members you anticipate having in the top-level data set. The value you provide for Avg No of Recs is the number of records you anticipate having in each member in the top-level data set. These two values are used to calculate the space required for the PDS or PDS-E at the top of the promotion tree. While these values are used to size the data set initially, you should make the numbers large enough to accommodate the anticipated growth of the data set. Figure 10 displays the completed Types tab for this example project.

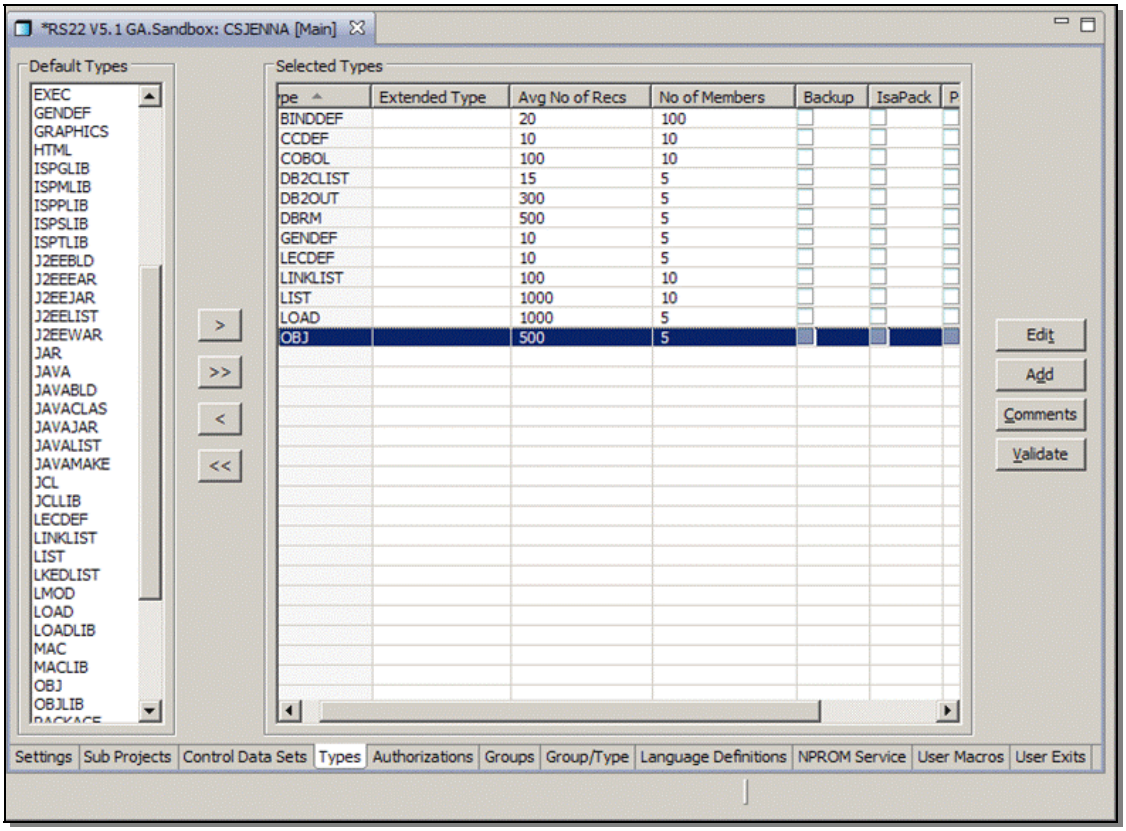

Figure 10: Define each data sets' parameters.

Other parameters that may be defined on this tab for each data set include Extended, Backup, IsaPack, Packfile and Reuse Days. The parameters available on this tab are defined below:

Extended Type: The name of a data set type in the project that contains members that should be included when the data set being created is processed by a language translator or architecture definition. For example, a COBOL data set type that contains COBOL source might have an EXTENDed type of COPY to include specific members from that data set type in the compilation.

The parameters Backup, IsaPack and Packfile are considered Package Backout flags and are mutually exclusive.

Backup: Check this check box if the members of this data set type are to be backed up during a promote. If you specify this check box, then at least one other data set type within the project must have the IsaPack option set, but they cannot be the same data set type. That is, the same data set cannot have both Backup and IsaPack set.

IsaPack: Check this check box if this data set type is to be used as the high level package file. That is, if this data set contains high level architecture definitions encompassing data sets whose contents are to be backed up, then select this option. As a result, when a promote is invoked on a high level architecture definition, all data set types referenced in the architecture definition are backed up.

Packfile: Use this check box to tell SCLM that this data set type contains the package file details. Only one data set type within a project can have the Packfile option set.

- **Reuse Days** (not visible in the screen shot)**:** This option only activates when the Packfile option is checked. Reuse Days specifies the number of days a package can be reused. SCLM checks the date of a package before a promote occurs. If the package being promoted is younger than the Reuse Day value, then SCLM reuses the package. If this package is older than the Reuse Day value, then the package details member is deleted and a new one created.

Note: This project does not use the Package Backout features of SCLM.

Now that the data set types have been added to the project definition, select the tab titled **Groups** in the Project Editor.

3.4 Define the Groups

SCLM has a concept called a hierarchy where the different stages of an application's source code are compartmentalized within separate sets of group/type data sets. The different stages of an application's development are called groups. Groups within SCLM are organized in a hierarchical order with each group being subordinate to the group above it when there is more than one group. The application's source code is developed in the lowest level of the hierarchy, and the completed application is cut from the highest level of the hierarchy, also known as the root.

An SCLM project hierarchy can be thought of as an upside down tree or a pyramid, where there can be many different development groups, and only one root group. Figure 11 illustrates a simple hierarchy in which there are two development groups and only one path through to the root group called PRD0110.

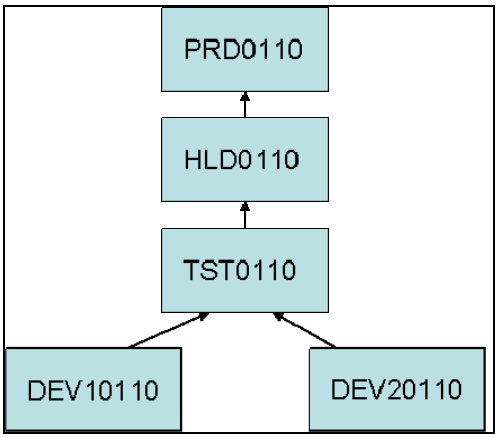

Figure 11: A sample SCLM project hierarchy.

Using the example in Figure 11 above, application source code can promote from one of two different paths: one path begins with code originating in group DEV10110, and one path begins with code originating in group DEV20110.

When a project hierarchy is defined within SCLM, the movement of an application's source code is controlled by which group is higher in the hierarchy. Therefore, the code from DEV10110 cannot be promoted to group HLD0110 until it has been promoted into group TST0110. Developed code can be promoted to the next group. The word "promote" means to copy or move a member or a set of members from one group to the next group in the hierarchy. Each group can only promote members to the group to which it is subordinate and to which the authorization codes match. This link between groups is known as the promotion path.

Using this concept, the tab titled Groups within the Administrator Toolkit graphical user interface allows you to specify the parameters for each group independently, as shown below in Figure 12.

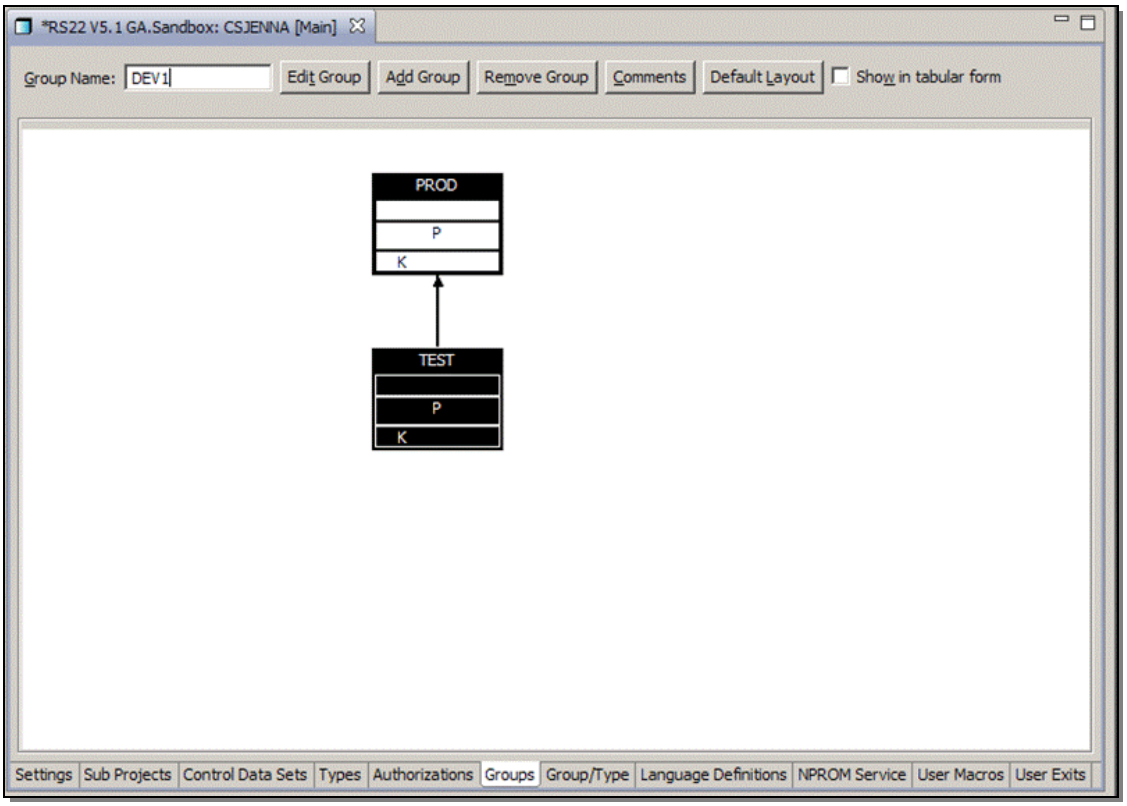

Figure 12: Specify the groups to include in your project's hierarchy.

Using the workstation client, you may drag and drop the group boxes around the display to map the hierarchy of the project. Once you have created all the groups you want to include within your project, you can create the hierarchy by right-clicking the lower group, then right-clicking the upper group. An arrow will appear indicating the direction of promotion. Using the example in Figure 12 above, the group called TEST will have its code promoted to the group called PROD. To create a new group to add to the project, simply type its name in the field titled Group Name, as shown in Figure 12, then click the button titled **Add Group**.

In Figure 13 below, the project hierarchy is completed, and shows the entire direction of promotion for the project in this case study. When you hover the cursor over a group, a pop-up window displays the group's parameter values. They may be edited by either double-clicking the highlighted group, or by highlighting a group name, and clicking on the button titled Edit Group toward the top of the Project Editor.

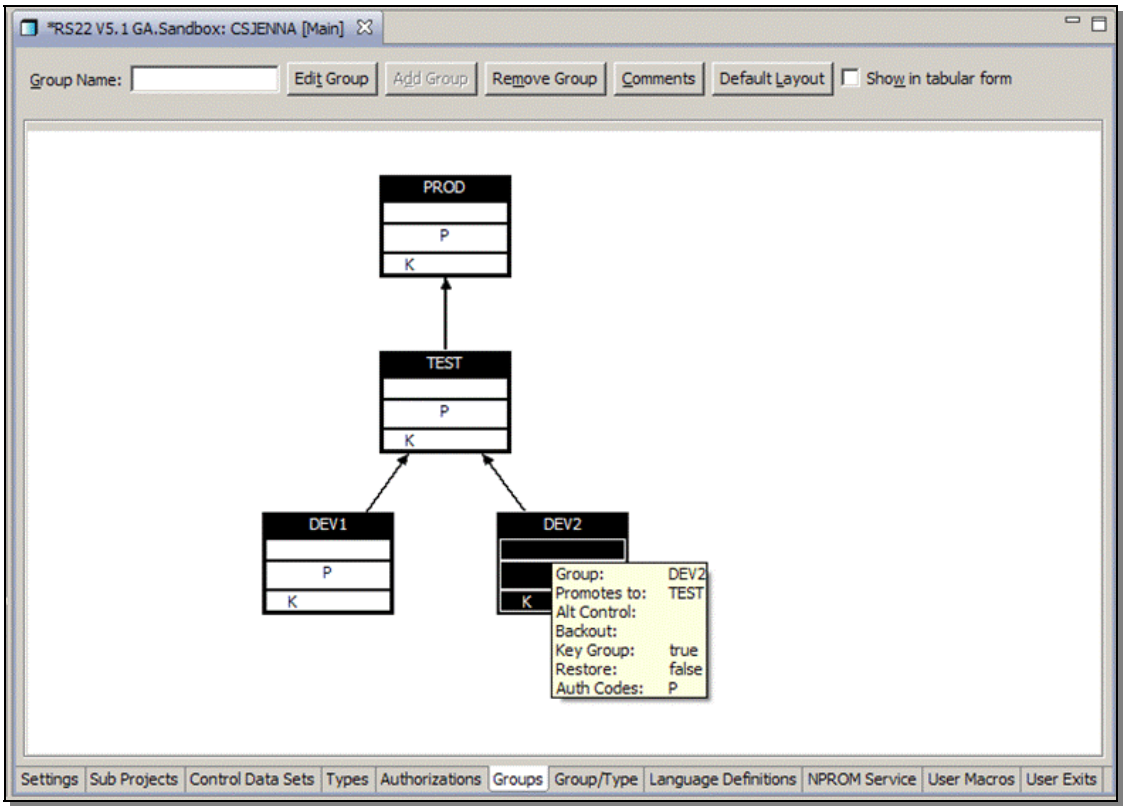

Figure 13: The completed project hierarchy showing the direction of promotion.

In Figure 14 on page 17 below, the Groups tab can display the project's groups in a tabular form, making it easy to see each project's parameters at-a-glance. You may sort the project group data within the table by clicking on any of the column headers. To edit a group's parameters, you may either highlight a group within the list and click on the button titled Edit Group, or you may place your cursor within the field whose data you want to change, and click on the drop-down arrow to select a value from the available values in the drop-down list.

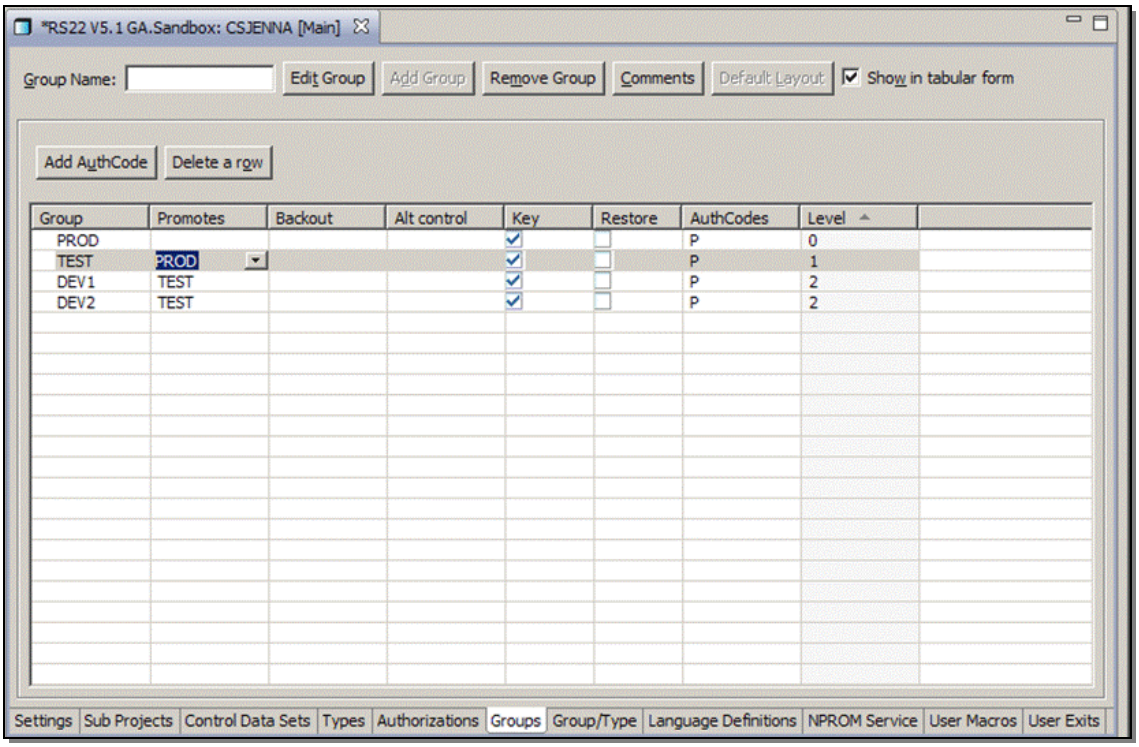

Figure 14: Tabular form displays each group's parameters in an easy-to-view list.

Now that the different data set types have been defined to the project, and the project groups have been defined to the project hierarchy, the Group/Type datasets can be selected and added to the project. In the Project Editor, click the tab titled **Group/Type** to continue.

3.5 Select the Group/Type Data Sets

Group/Type data sets are those data sets that consist of the project group name and the data set types added from the Types tab. Simply put, they are data sets whose names are derived by combining the project group name and data set type. On the Group/Type tab, all possible data set combinations are presented within a list for you to choose which ones to include within your project. The reason why you have the ability to pick and choose the group/type data set combinations to add to your project is because perhaps not all development groups are to develop assets in the same language. Perhaps one development group is to develop COBOL assets while another development group is to develop assembler assets, as shown in Figure 15 on page 18.

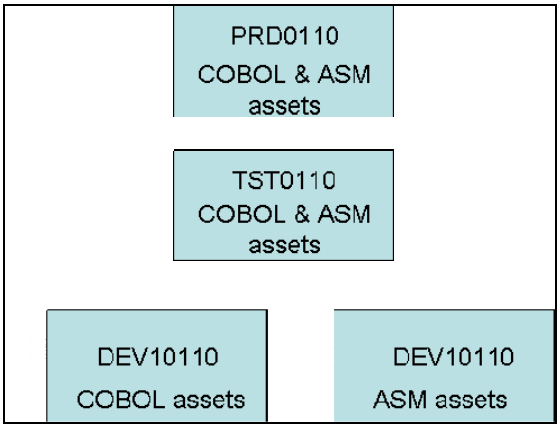

Figure 15: Sample hierarchy illustrating different groups requiring different kinds of data sets.

In this example, group DEV10110 will contain COBOL code and would not require ASM type data sets allocated. Group DEV20110 will contain ASM code and would not require COBOL type data sets allocated. However, the groups that they promote into would require the same kinds of data sets allocated for the promotion to complete successfully.

On the Group/Types tab of the Project Editor, there are two frames listing different group/type data set combinations, as shown in Figure 16. The available Group/Type Combination data sets displayed in the frame on the left are created by pairing the defined groups on the Groups tab with the data set types selected on the Types tab. The frame on the right includes Group/Type Allocations that you select to be included in the project.

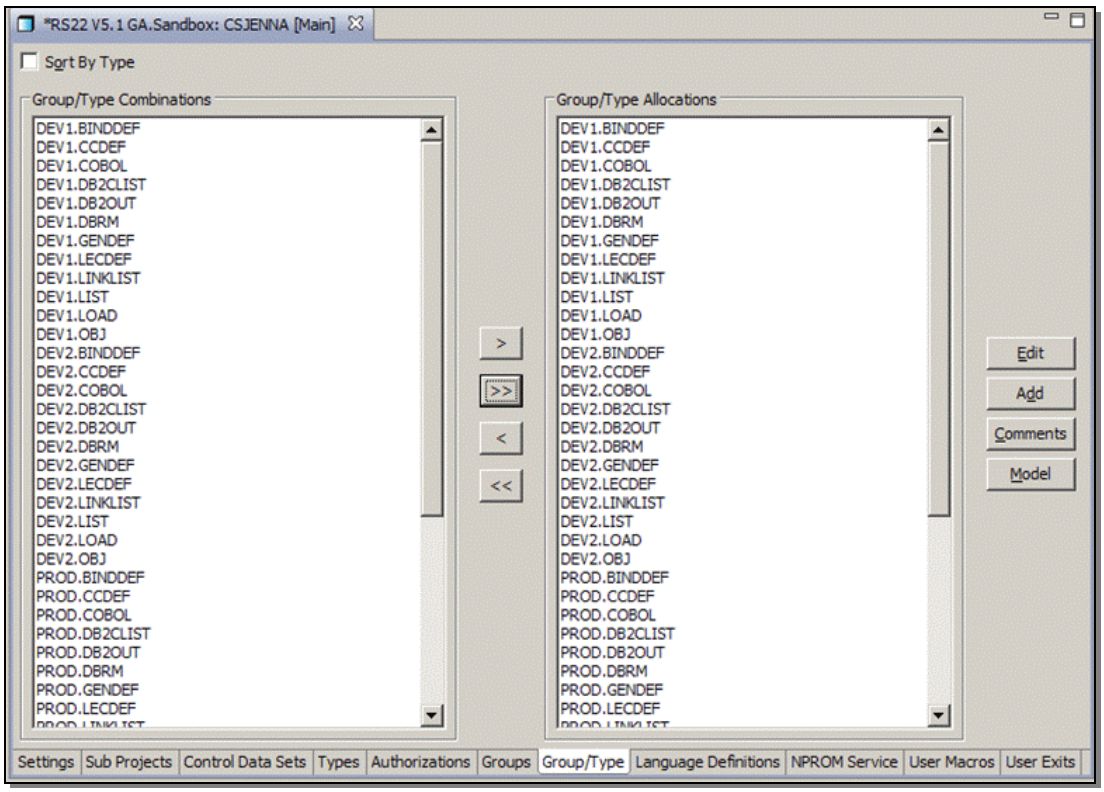

Figure 16: Select the group/type combination data sets to include in your project.

You can pick and choose which data sets to include in your project based on your project's requirements by highlighting the group/type data set and clicking the single right arrow to move it to the right-hand frame. You may select multiple data sets at a time by holding the CTRL key while you click on each data set, then click on the single right arrow. You may optionally move all data sets from the left hand frame to the right-hand frame by clicking on the double right arrow.

If you are unsure if you have selected the correct Group/Type combination data sets within the hierarchy to accommodate lower-level development groups, you can use the Validate Project function from the SCLM option on the menu bar to notify you what data sets, if any, are missing within the hierarchy. You can then add those that are missing by selecting the data set from the list in the left-hand frame.

For this example, all group/type data sets will be used within the project. Therefore, clicking the double right arrow will move all group/type data set combinations to the right-hand frame.

No project is complete without language definitions that dictate how source code is translated into object modules, or load modules. To add language definitions to your project, click on the tab titled **Language Definitions** within the Project Editor.

3.6 Add Language Definitions

Language definitions are translators that process an application's source code from input to output, or to another form of input. For example, it can translate an application's source code into object code and listings, and then translate the object code into load modules. Because of their complex nature, new language definitions can be difficult to fine-tune. To help the user establish new projects, SCLM provides many language definitions that you can use in your project "as-is" or with minor modification.

You might also have custom languages or third-party languages you want to use within your project but getting them to run properly within SCLM might be difficult. To overcome these inhibitors, the Administrator Toolkit has a separate utility called the Language Definition Wizard that can assist users in creating a new language in less time and with fewer errors. Using context-sensitive fields and default parameters to help new SCLM administrators through the creation process, users are notified of conflicting parameters so a change can be made before the project is put into production.

On the Language Definition tab, new projects will have an empty list of languages. If you clone an existing project's definition, the list of languages copied from the existing project will be displayed.

In our example, the Language Definition tab will be blank. To the list, we will add languages from the SCLM macro library. Some will be copied "as-is" with no edits made to the language, and some will be copied, but edited for the purpose of our project.

When you click the button titled Add, the Language Definition Wizard is invoked and allows you to specify how you want to add a new language definition. As shown in Figure 17 below, you can add a language definition in one of three ways.

Create from an existing Language Definition Data Set: Select this radio button if you simply want to copy an existing language definition. If the option 'Create as is' is not checked, then a copy of the language definition is placed within your project's SOURCE data set, and you are allowed to edit its parameters. If the option 'Create as is' is checked, then the language definition remains in its original data set and a COPY statement is placed within your project's SOURCE data set referencing the original data set in which the language resides. You will not be able to edit its parameters when 'Create as is' is selected.

Create from existing Language Definition JCL: If you have third party languages, or if you have existing JCL that translates an application source code from input to output, or to another form of input, use this option to create a Language Definition. The existing JCL is parsed to create a preliminary language definition in which you can then edit and tailor it for your project's needs.

Create by providing the Language Definition parameters: You can create a language definition from scratch by using the Language Definition Wizard to guide you.

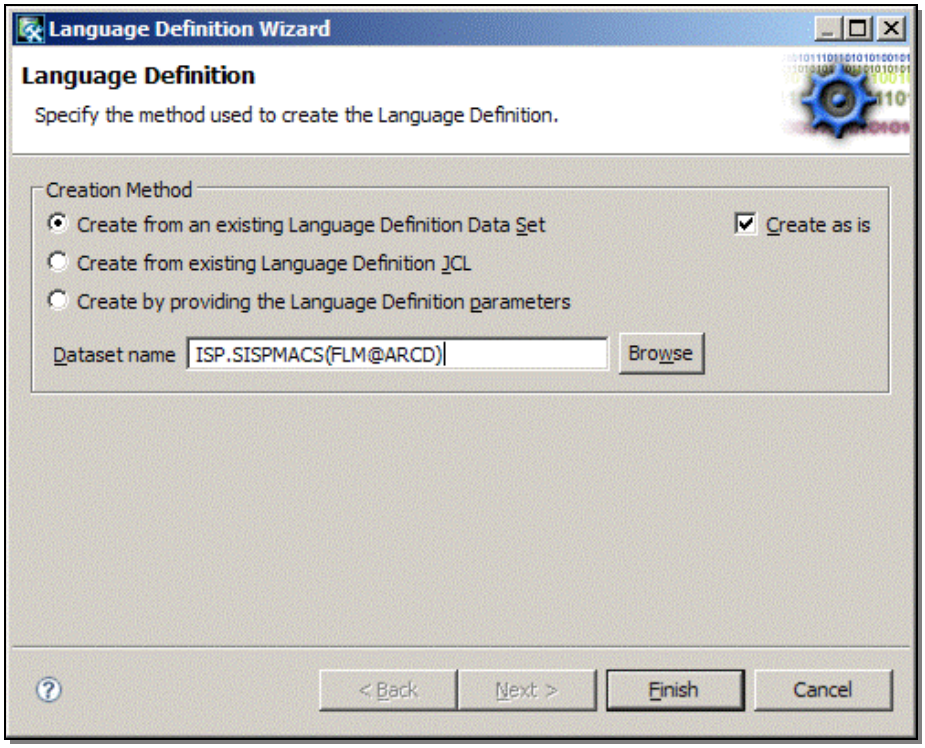

Figure 17: Select the option you wish to use to create a Language Definition.

With the exception of the option 'Create as is', the Language Definition Wizard will guide you through the process of creating your language definition. Content-sensitive fields prevent you from entering conflicting parameters, helping you to understand why the parameters conflict and assisting you in resolving the conflict. Field-level help is available throughout the Administrator Toolkit, but becomes invaluable when you need additional information to help resolve a conflict.

In our example, the language FLM@ARCD, an architecture definition language, is being copied ‘as-is’ from the SCLM macro data set. Because you cannot edit any parameters from a language created ‘as-is’, the language definition is copied into the project definition but is referenced from its original data set. A message indicating a successful copy is displayed, as shown in Figure 18.

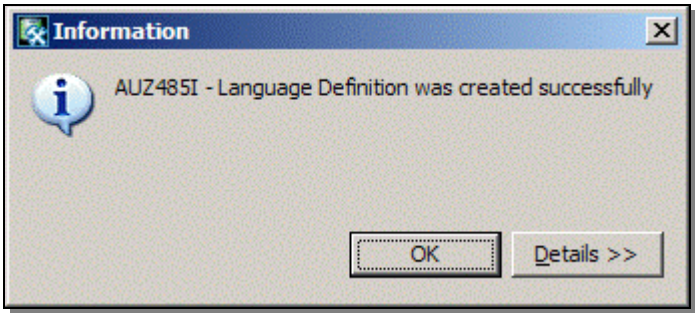

Figure 18: A message is returned when a language is successfully added to a project.

In our example, the following languages are added to the project: FLM@ARCD, FLM@BD2, FLM@BDO, FLM@L370, and FLM@2CO2. Languages FLM@L370 and FLM@2CO2 are edited to reflect changes that are needed for this specific case. To illustrate how the Language Definition Wizard assists users with editing language parameters, edits to language FLM@L370 are displayed below, beginning with Figure 19.

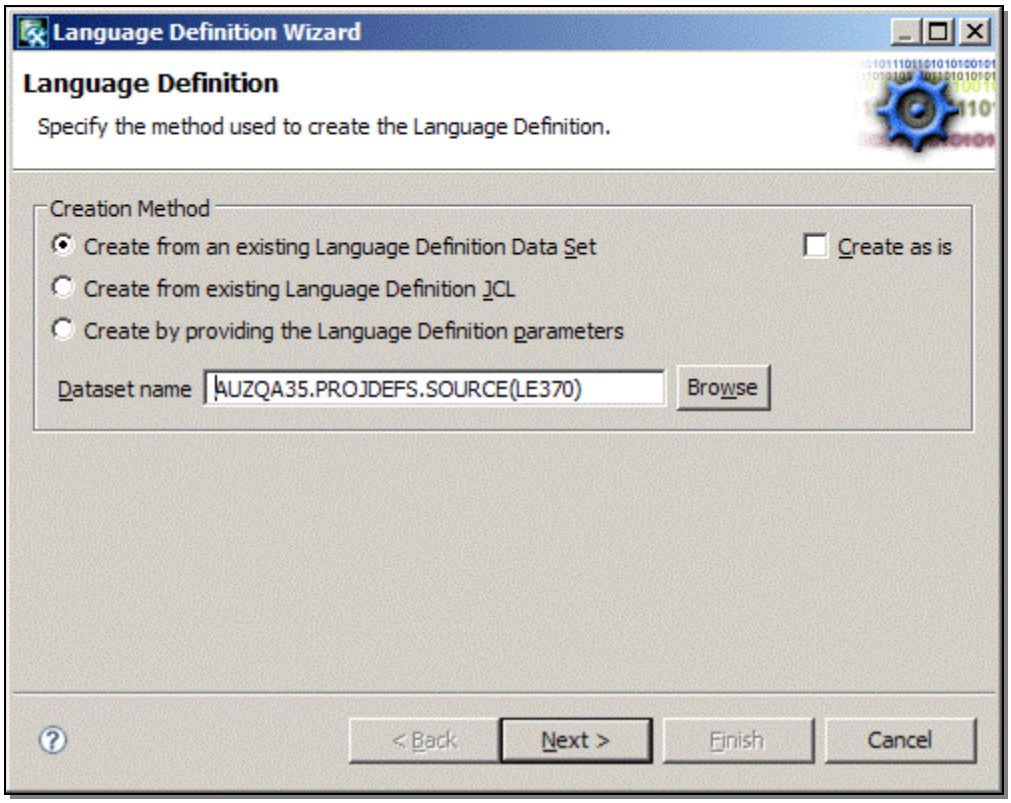

Figure 19: Specify a language to copy, but edit the language to tailor it to your project.

You may copy a language from any data set, not just the SCLM-provided macro data set. Within the field titled Dataset Name, provide the data set name and member name that

contains the language you want to copy. Click the radio button next to the option titled ‘Create from an existing Language Definition Data Set. Then click **Next**.

You may optionally search for the data set name by providing a partially qualified data set name and/or member name, and clicking the button titled Browse. When you do so, a window titled Dataset Selection Dialog will display, as shown in Figure 20, with a list of data sets whose name matches the pattern you specified. You can then click on the data set name to expand its member list, as shown in Figure 20 below. Simply select the member name that is the language definition you want to copy. Then click on **OK**.

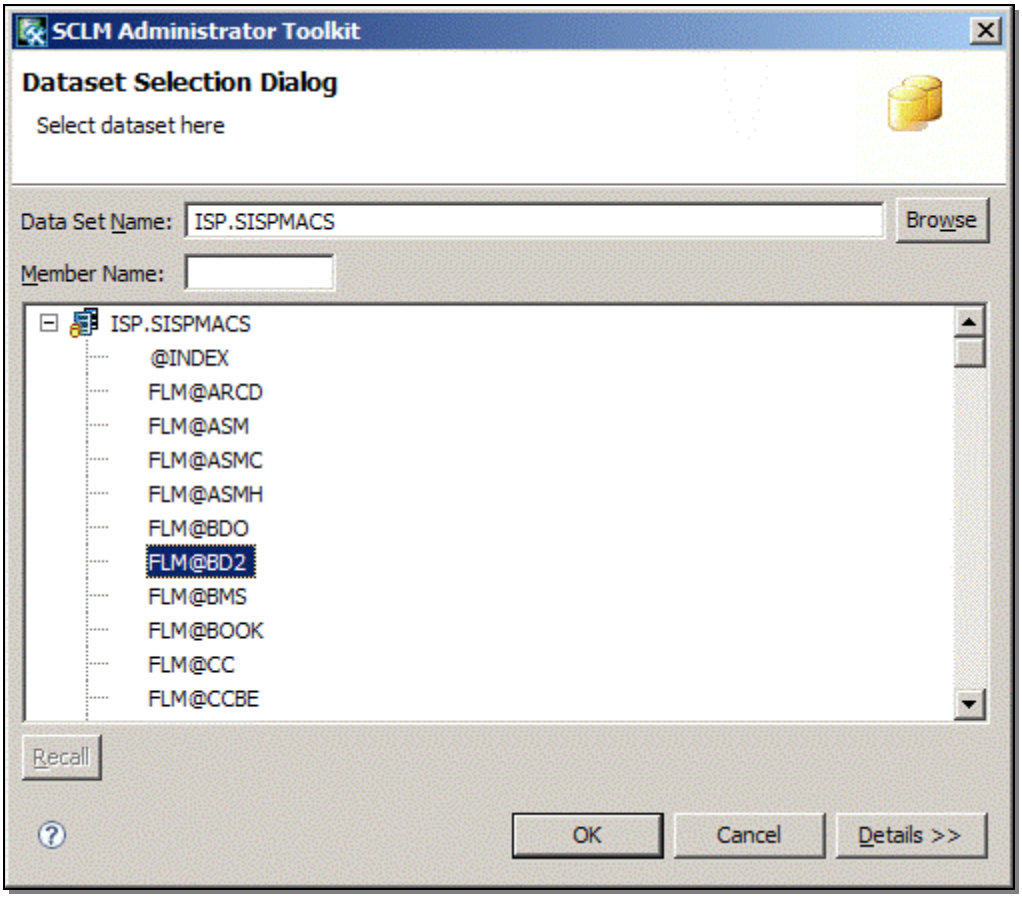

Figure 20: Expand the data set name and select the member name to use.

When you select the data set and member name that contains the language definition you want to copy, you are returned to the main Language Definition Wizard panel. Notice that the field titled Dataset Name is completely filled in. Simply click **Next** to continue.

The parameters for the FLMLANGL macro are displayed for your review, as shown in Figure 21. You may edit any of the parameters that are displayed to tailor it to your project’s requirements.

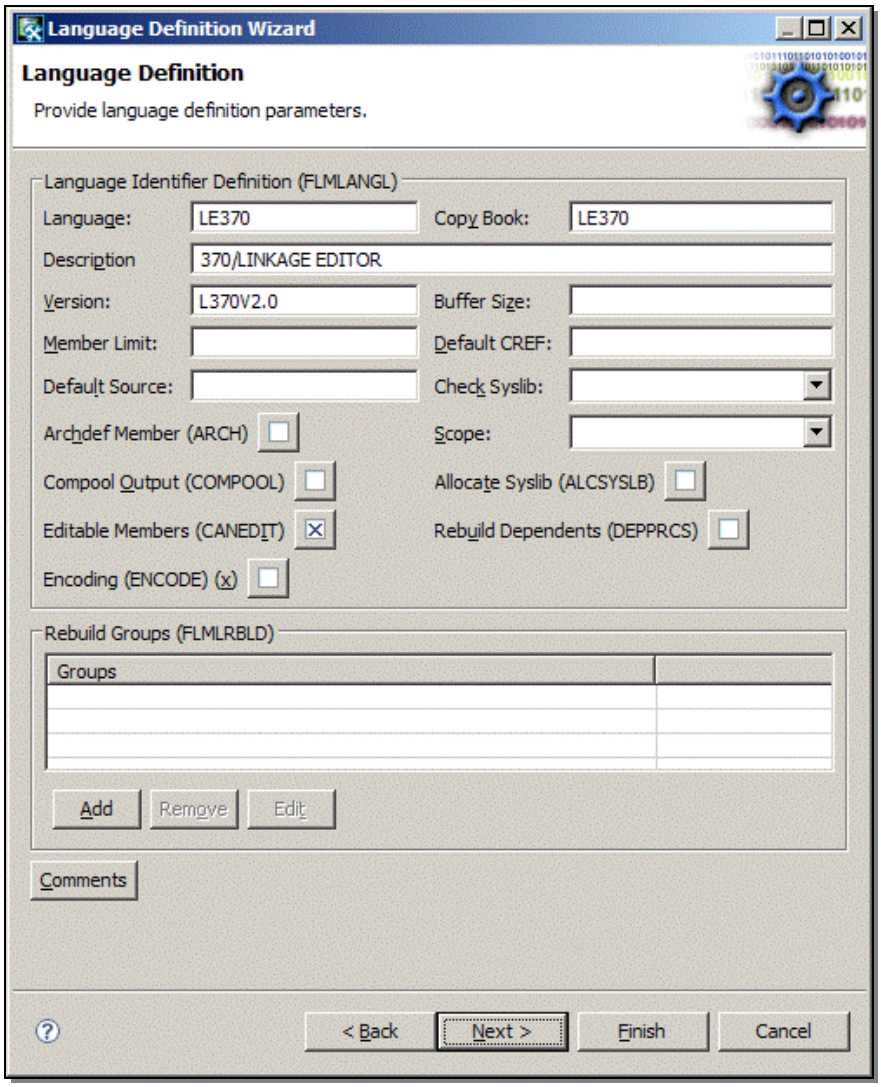

Figure 21: Make any necessary edits to the language definition to tailor it to your project.

Clicking **Next** will advance you to each of the macros and their corresponding parameters. In this example, the translator parameters will be edited. The Translator Summary window, as shown in Figure 22 on page 24, displays all translators that are defined within a language definition. Some languages have many different translators. However, the link-edit language definition only has a BUILD translator.

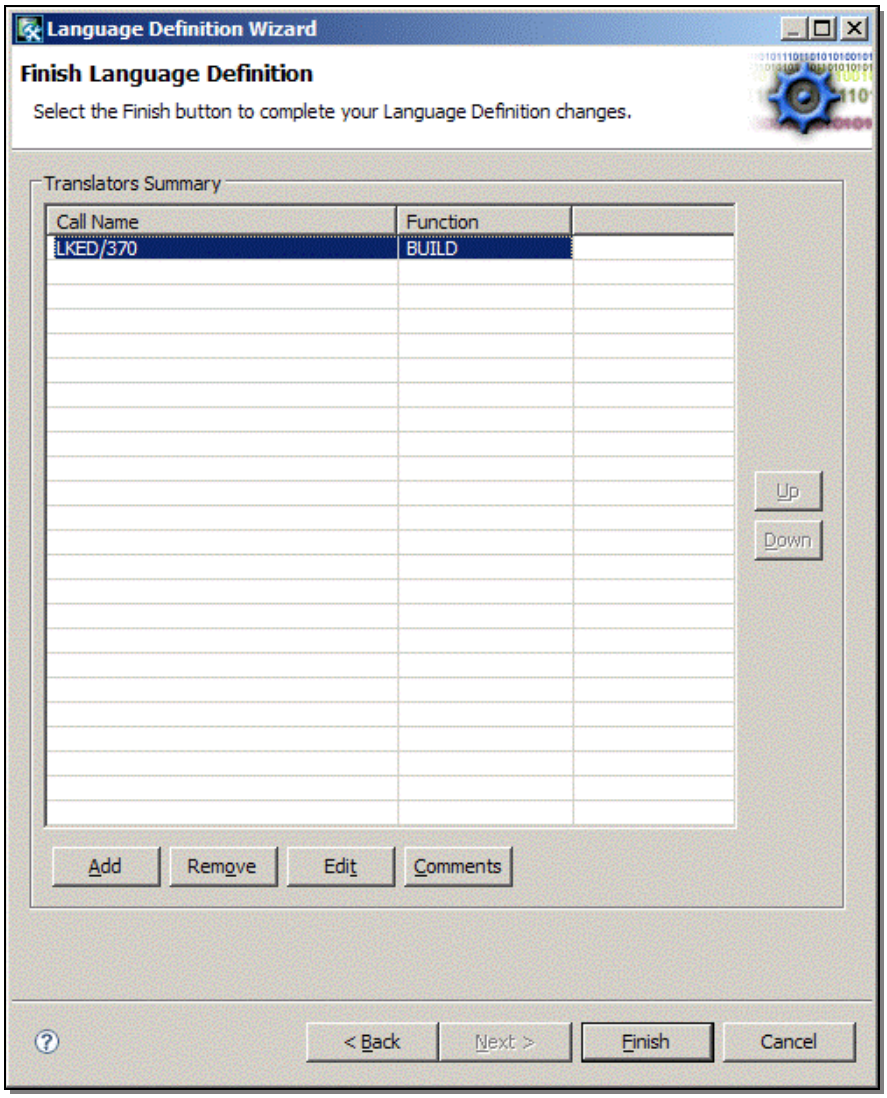

Figure 22: The Translator Summary provides a list of translators defined within the language.

To make any necessary edits, simply highlight the translator within the list, and click the button titled **Edit**. A window titled Translator Parameters will display, allowing you to edit FLMTRNSL macro parameters within the BUILD translator, as shown in Figure 23 on page 25.

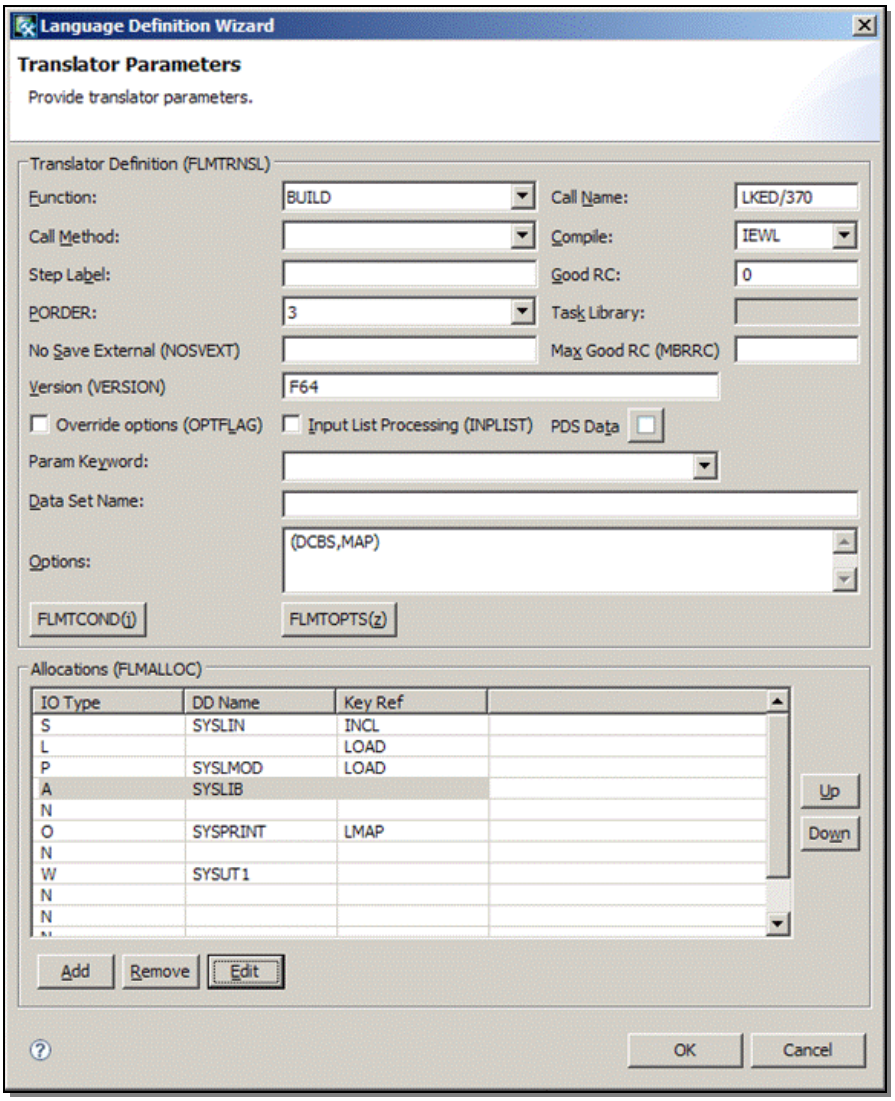

Figure 23: All FLMTRNSL macro parameters can be edited.

Within this example, the FLMALLOC data sets will be edited to modify the data sets associated with the SYSLIB data set. Highlight the SYSLIB data set within the frame titled Allocations, and click **Edit**. A window titled Data Set Allocation Dialog will allow you to edit the FLMALLOC macro parameters for the SYSLIB data set, as shown in Figure 24 on page 26.

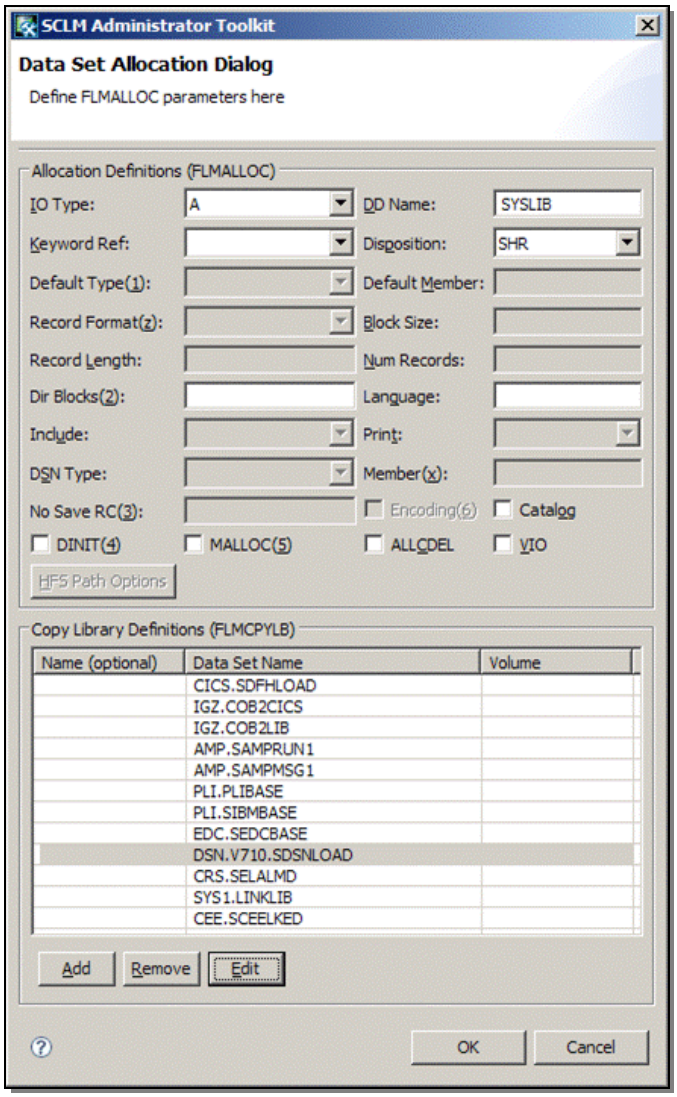

Figure 24: Edit the data sets concatenated within the FLMCPYLB macro.

In our example, the DB2 LOAD data set must be edited to reflect the correct version of DB2 that the project will be executed on. To edit a data set name within the list, highlight the data set, and click **Edit.** A pop-up window titled Copy Library Definitions will display allowing you to edit the data set name, as shown in Figure 25.

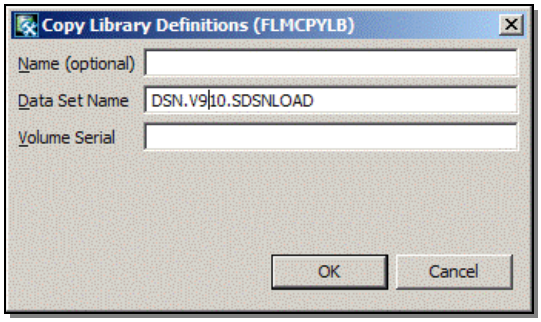

Figure 25: Edit the Copy data set name as necessary.

When you are done editing the data set name, simply click **OK** to return to the Data Set Allocation Dialog to view your changes. Click on **OK** to continue editing the language definition. On the window titled Finish Language Definition, click **Finish** to complete the Language Definition Wizard, and return to the Language Definitions tab within the Project Editor, as shown in Figure 26.

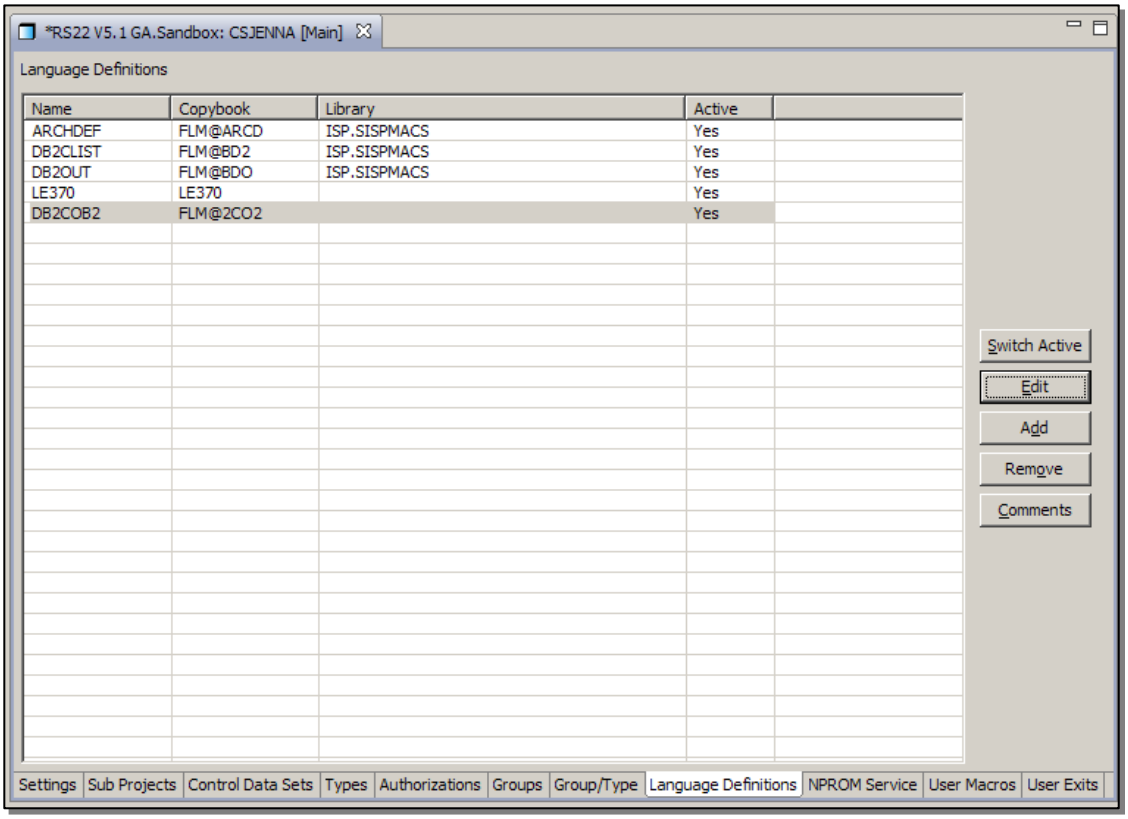

Figure 26: The list of language definitions is displayed within a list.

Notice that languages LE370 and DB2COB2 are not referenced within a library, while the other three languages, such as ARCHDEF, are referenced in library ISP.SISPMACS. This is because the languages ARCHDEF, DB2CLIST and DB2OUT were created “as-is”. Remember that creating a language “as-is” simply places a reference to the original language definition within your project. It does not copy the language definition to your project.

In this example, a copy of language LE370 is placed within this project’s definition data set because an edit was made to the language. To prevent updates from being made directly to the ISP.SISPMACS library, where few users will have update authority, a copy is placed within the project’s definition data set, and any usage of that language by the project’s application source code will come from the copy that was placed within the project’s definition data set.

3.7 Customize the DB2 CLIST

Because this sample project is a DB2 project, a DB2 CLIST must be included and customized as part of the project’s definition. A DB2 CLIST is either a TSO CLIST or REXX exec procedure that allows you to BIND or FREE plans or packages for each DB2 program within your application.

Select the tab titled **Settings** within the Project Editor. The Administrator Toolkit can automatically create this CLIST for you using the button titled DB2 Management on the Settings tab, as shown in Figure 27.

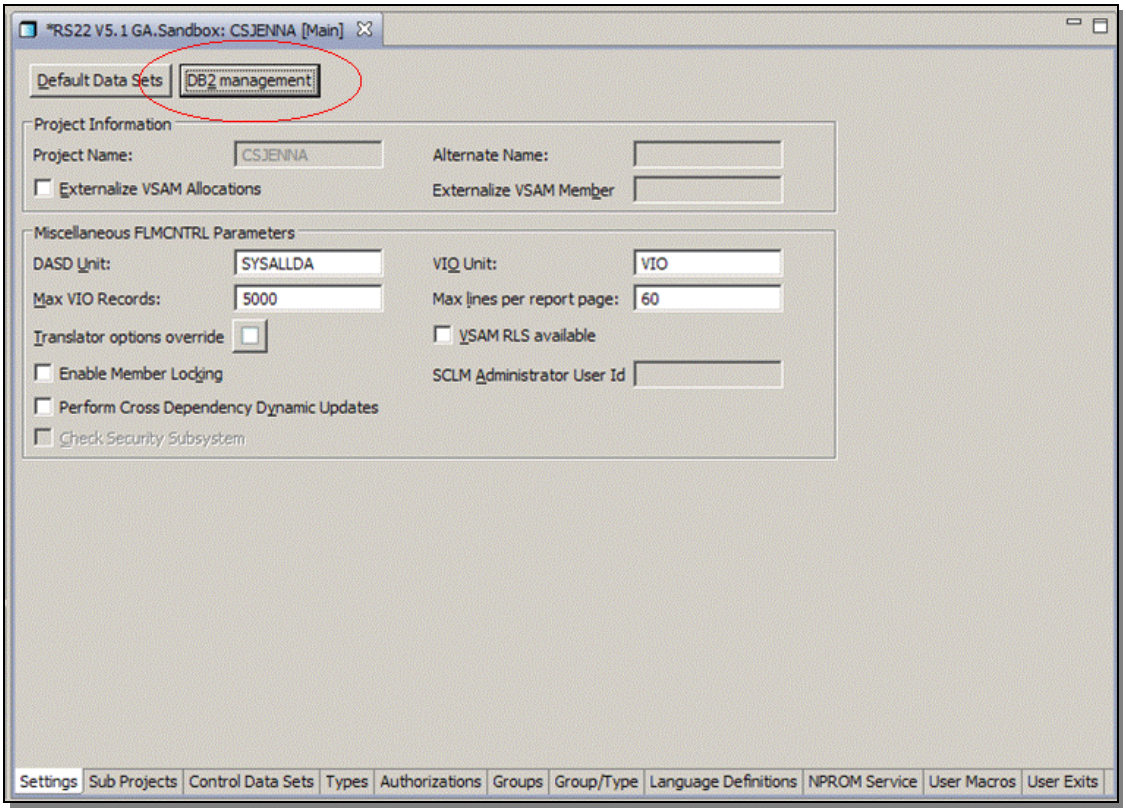

Figure 27: Automatically create the DB2 CLIST which drives the plan or package BIND process.

When you click on the button titled **DB2 Management**, a window titled DB2 BIND Parameter Defaults is displayed, as shown in Figure 28, which allows you to specify the parameters to use for BIND syntax.

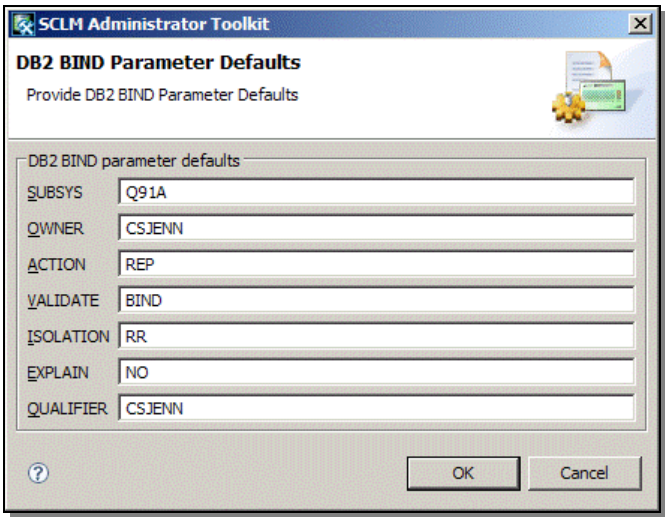

Figure 28: Provide BIND syntax parameters for the DB2 applications within your project.

Enter all the required parameters for your DB2 program within the appropriate fields. For more information on the definition and usage of each of the DB2 BIND parameters, please refer to the chapter on the BIND PLAN and BIND PACKAGE syntax within the IBM DB2 Command Reference Manual for the appropriate version of DB2 for which you are creating BIND syntax. Click **OK** when all BIND parameters required for your DB2 program have been entered.

3.8 Build the Project

When a project's definition is complete, and you have specified all options you want to include within the project, the project then has to be built. Building a project simply means that all SCLM macro parameters and their values are assembled and linked into an executable load module. It is this executable load module that SCLM uses when you invoke SCLM to access your project's source code and other project-related assets. If no load module exists for your project, then SCLM issues an error message informing you that the project's load module cannot be found.

To build your project's definition in the Administrator Toolkit, click on the menu bar above, and select the option titled **SCLM**, while your project is still open within the Project Editor, as shown below in Figure 29.

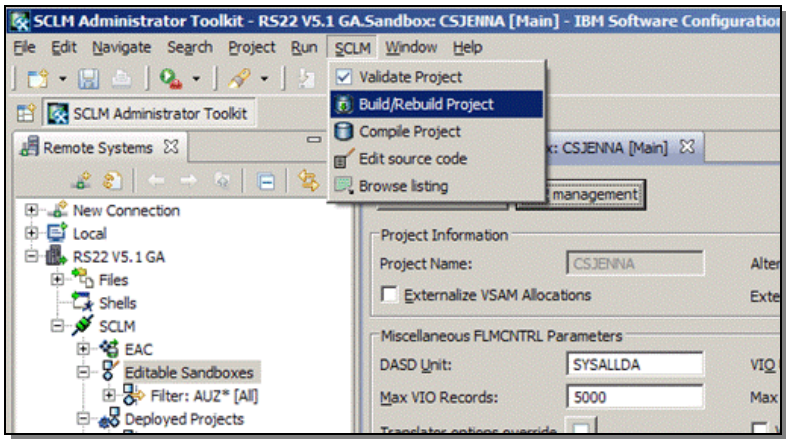

Figure 29: Select the option to Build your project.

Select the option titled **Build/Rebuild Project** from the drop-down menu. When you do, a window titled Build Project Dialog will appear, as shown in Figure 30 on page 30. The checkbox next to the option titled Create/update project environment is selected by default.

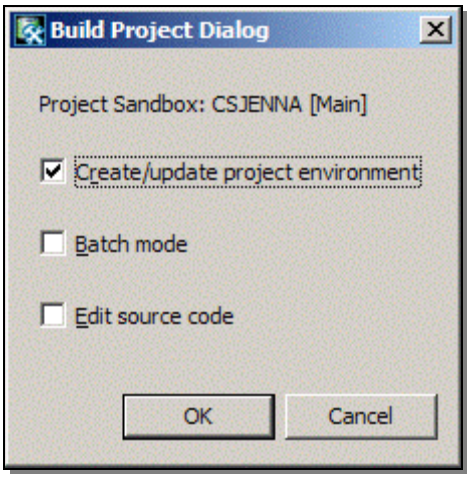

Figure 30: Click OK to build your project's definition into a load module.

Click **OK** to begin automatically building your project's load module. Within the lower-right hand side of the application window is a status indicator, as shown below in Figure 31. As the project is built, the status indicator increases providing you information on the tasks' progress.

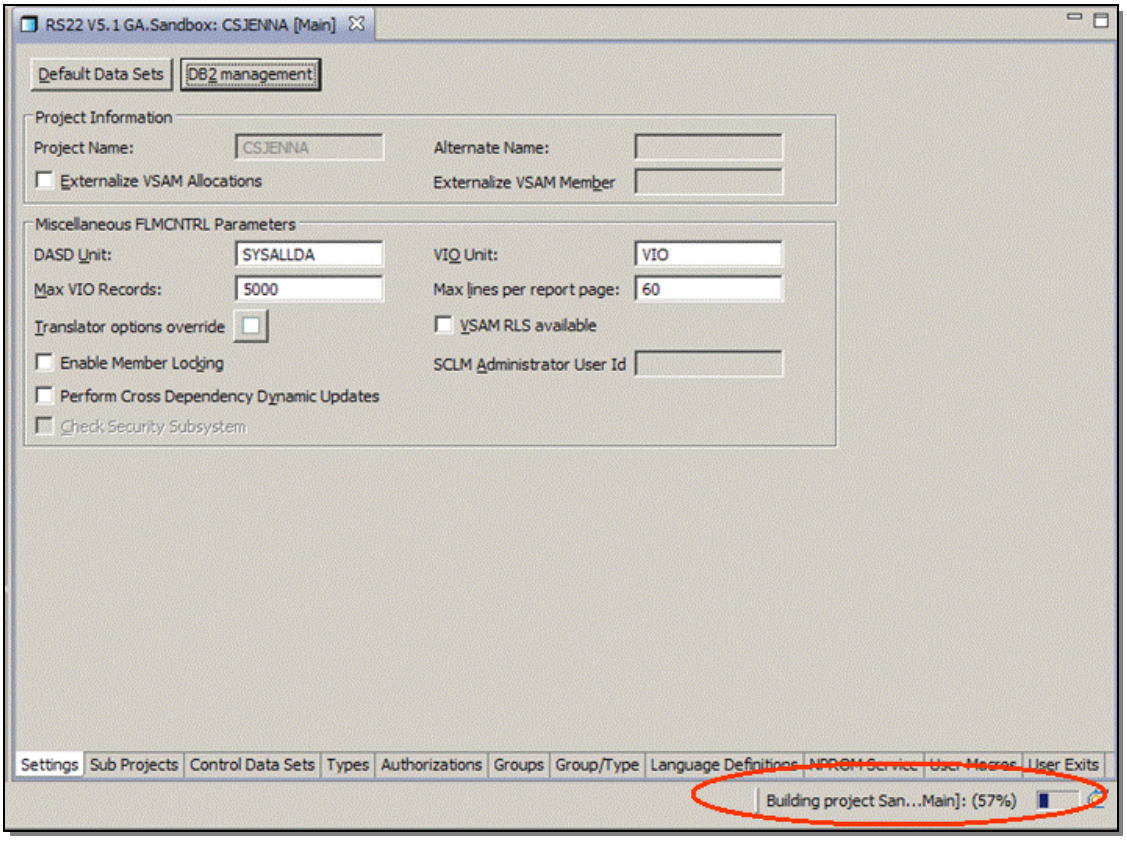

Figure 31: The status indicator provides a progress report of the current task.

When the project build has completed, a message will display along with any additional information related to the building of the project, as shown below in Figure 32.

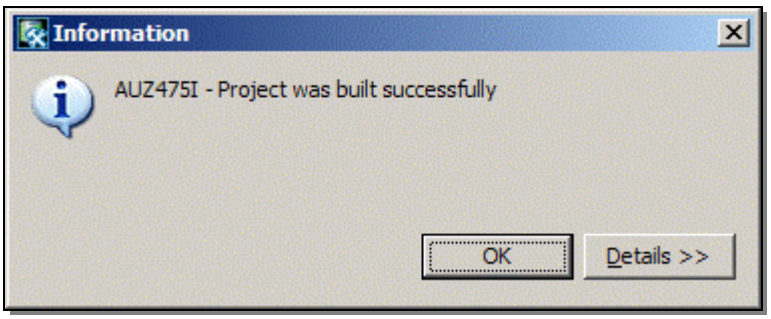

Figure 32: Messages are displayed within a pop-up window with additional information.

Clicking on the button titled Details will result in expanding the Information window to display additional information about the message. Click **OK** to close the informational message window.

Troubleshooting: *Should an error be returned during any task, the task is written to the Reports node within the Remote Systems view, as discussed in this document in the subsection titled* A Brief Introduction*. It is these reports that Product Support will request when opening a PMR with IBM for diagnosis.*

3.9 Create a Project Filter

In order to view your newly created project within the list of projects in the Remote Systems View, you must create a Project Filter whose pattern matches the name of the project you just created. To begin creating a project filter, right click on the node titled **Editable Sandboxes** within the Remote Systems view, as shown in Figure 33.

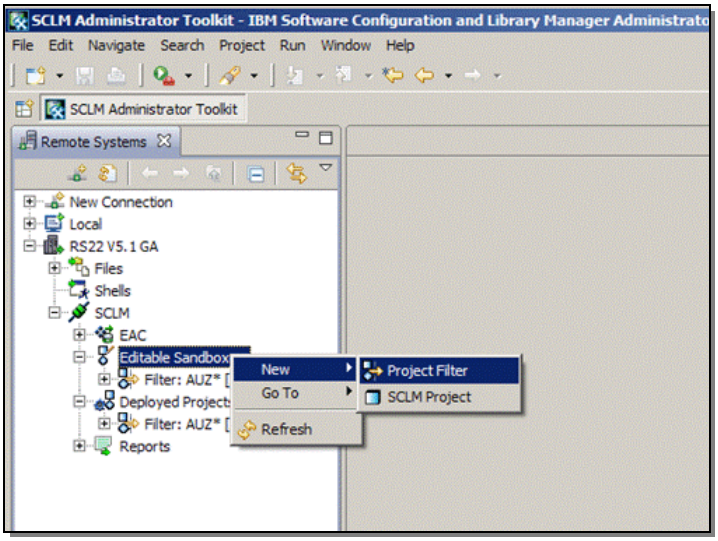

Figure 33: Create a project filter to view your newly created project.

An additional menu will display providing additional functionality. Select the option titled **New**. Then select **Project Filter**. When the window titled New SCLM Project Filter is displayed, provide a fully-qualified project name, or a partially qualified pattern using an asterisk as a wild card character, as shown below in Figure 34.

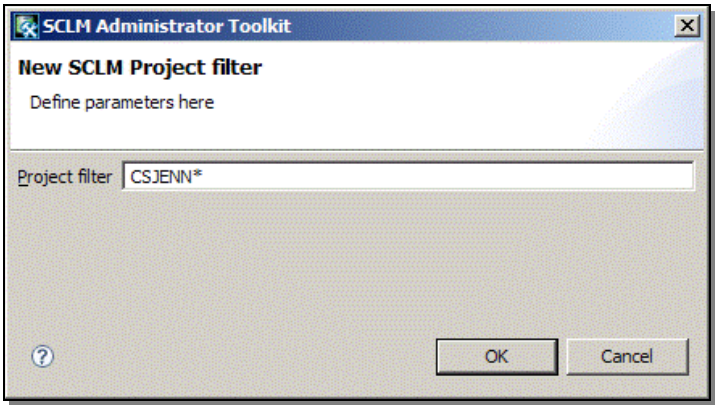

Figure 34: Provide the project filter criteria.

In this case study, the new project name was CSJENNA. To view a list of projects whose names match the specified criteria, an asterisk was specified as a wild card character. As a result, all projects whose name matches the specified criteria will be displayed when the project filter is expanded. Click **OK** when you have completed entering the project filter criteria.

Once you have created a new project filter, the node titled Editable Sandboxes must be refreshed to see the changes made to the available project filters. **Right click** the node Editable Sandboxes. Then click **Refresh**, as shown below in Figure 35.

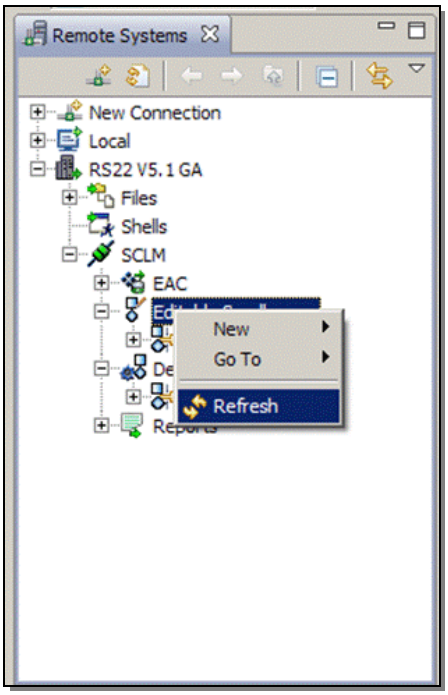

Figure 35: Refresh the node to view the new project filter.

When the node Editable Sandboxes is refreshed, the project filters are displayed, as shown below in Figure 36. Click on the **plus sign** ('+') next to the project filter to expand the filter and view the list of projects whose name matched the filter criteria.

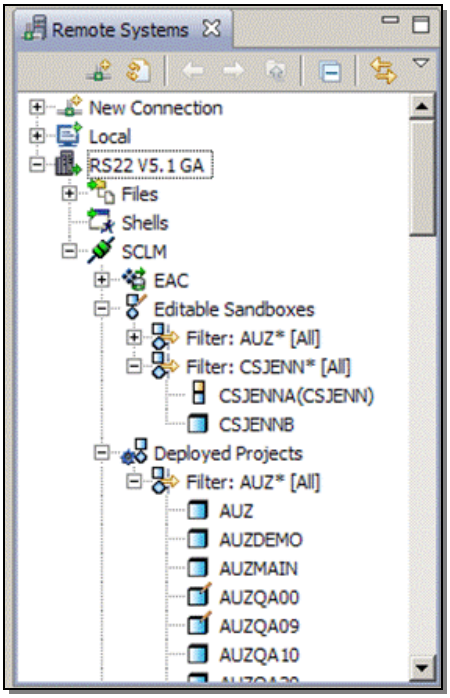

Figure 36: Expand the new project filter to see all projects that match the filter name.

In this case study, there are two projects that match the project filter criteria; the project named CSJENNB was a pre-existing project that is not covered within this case study. Project CSJENNA is the project sandbox in which our edits were made. However, the real project is called CSJENN. Remember that when we were initially creating the project, we named the project CSJENN, while the sandbox was called CSJENNA. Refer to the section within this document titled Create a new project on page 3 to review the steps we took to create the project and its corresponding sandbox.

The icon next to Project CSJENNA has two small squares; one yellow and one blue. This icon means that CSJENNA is a sandbox project, and the project name within parenthesis is the main project, CSJENN. Remember that a sandbox is simply a project contained within a separate set of libraries from the main project, which allows you to test and tweak the sandbox project without affecting the main project.

The icon next to project CSJENNB is a blue square, which means this project is being actively used within SCLM.

The icon next to project AUZQA00 is a blue square with a small pencil. This project is under the node titled Deployed Projects, which means it is actively used in production, but is editable.

Now that the project CSJENNA has been successfully built, application source code from another project can be copied, or migrated, into the project. In this case study, we will migrate a DB2 program into this project's group/Type data sets for use in our project.

3.10 Migrate Project Assets

Software assets can be easily migrated into an SCLM project from existing libraries using the Migration Wizard or from remote systems using the Remote Migration Wizard. The Migration Wizard and Remote Migration Wizard guide you through migrating assets into your project by using content-sensitive fields and field-level help. Assets, including those with long file names, can be migrated into SCLM projects and the corresponding accounting data is automatically updated for each asset migrated into your project.

Migrating assets allows you to use application source code from another project that might contain a function your product has, or you can copy code from non-SCLM-managed libraries into an SCLM-managed project. These are just two examples of why migrating assets into a project is sometimes necessary. Utilizing the Migration Wizard in the Administrator Toolkit helps ensure the assets are migrated correctly, and automatically updates project control files, such as the accounting and auditing files.

To invoke the Migration Wizard, right click the project into which you want to migrate assets. When the pop-up menu is displayed, select the option titled Migration Wizard, as shown below in Figure 37.

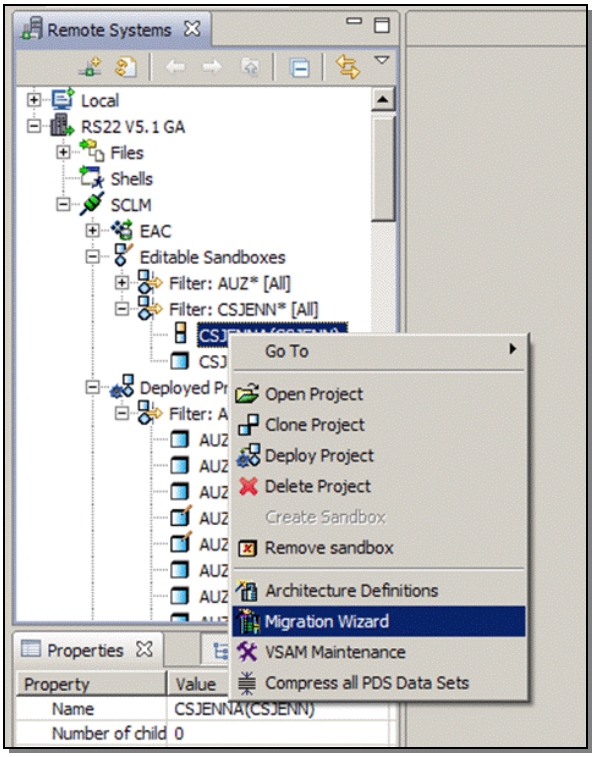

Figure 37: Right click your project to navigate to the Migration Wizard.

When the window titled Migration Wizard is displayed, it will initially be blank, as shown below in Figure 38. You may specify as many assets to migrate at one time only if they will use the same language definition.

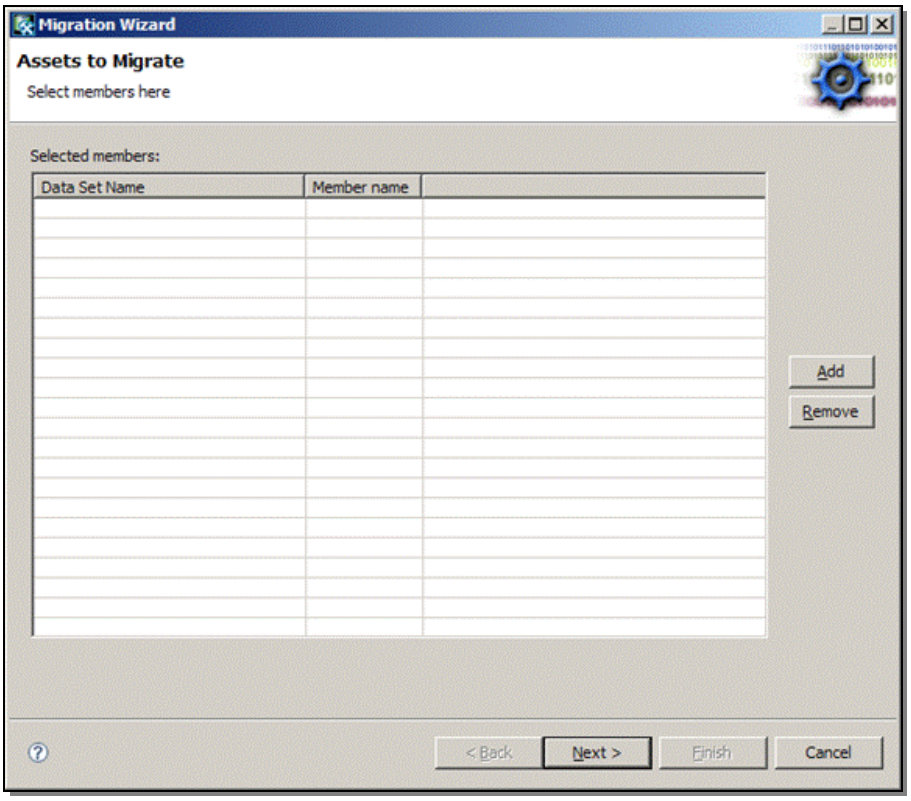

Figure 38: Click the button Add to add an asset to the list.

Click the button titled Add on the right-hand side of the window. A window titled Dataset Selection Dialog will display allowing you to specify the file that contains the assets you want to migrate. You may partially specify the data set name using an asterisk ('*') as a wild card character, as shown below in Figure 39 on page 36. Likewise, you may fully qualify the data set name and optionally its member name.

Click the button titled Browse to search for the data set name if you wild carded the data set name.

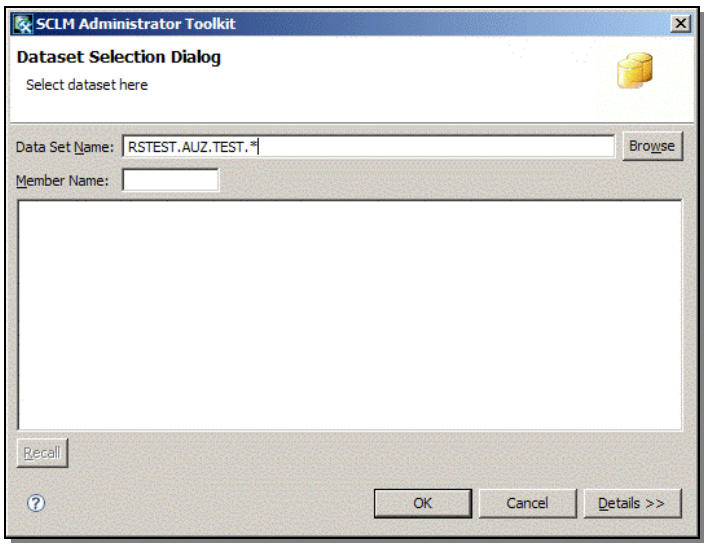

Figure 39: Specify the data set that contains the assets you want to migrate.

The Migration Wizard will search for all PDS data sets on the host on which your project resides. The data sets that match the specified data set name or pattern are displayed in the bottom frame. Select the data set that contains the assets you want to migrate, as shown in Figure 40.

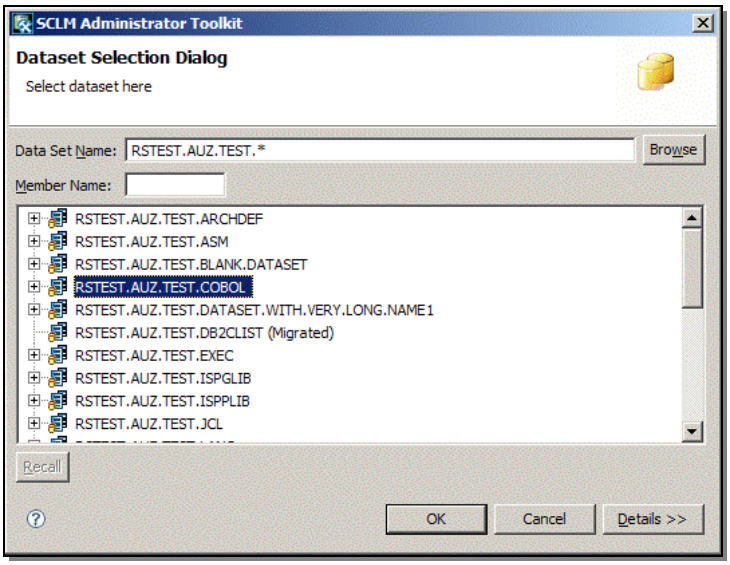

Figure 40: Select the data set that contains the assets to migrate.

When you click on the plus sign ('+') next to the highlighted data set name to expand the member list, all members contained within the data set are displayed, as shown in Figure 41.

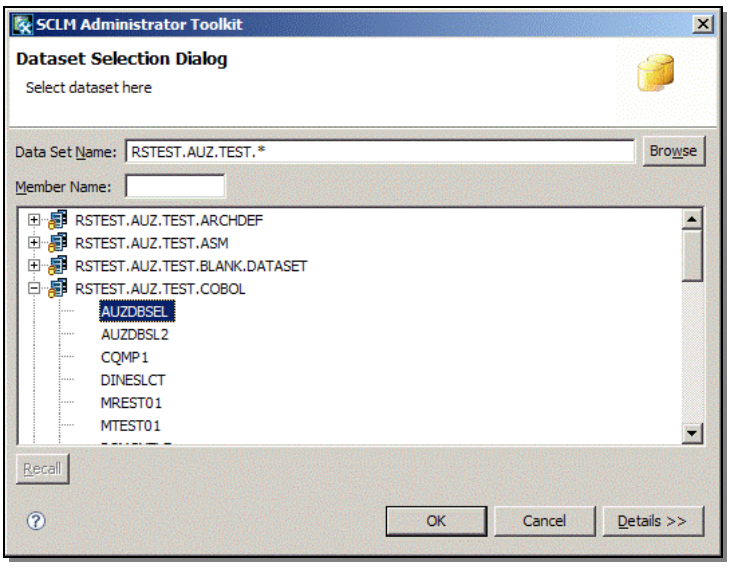

Figure 41: Select the members you want to include in the list.

Select multiple members using the CTRL key and the left mouse button. Click **OK** when you have completed selecting the assets you want to migrate. The Migration Wizard window is re-displayed, listing all assets you selected to migrate, as shown below in Figure 42.

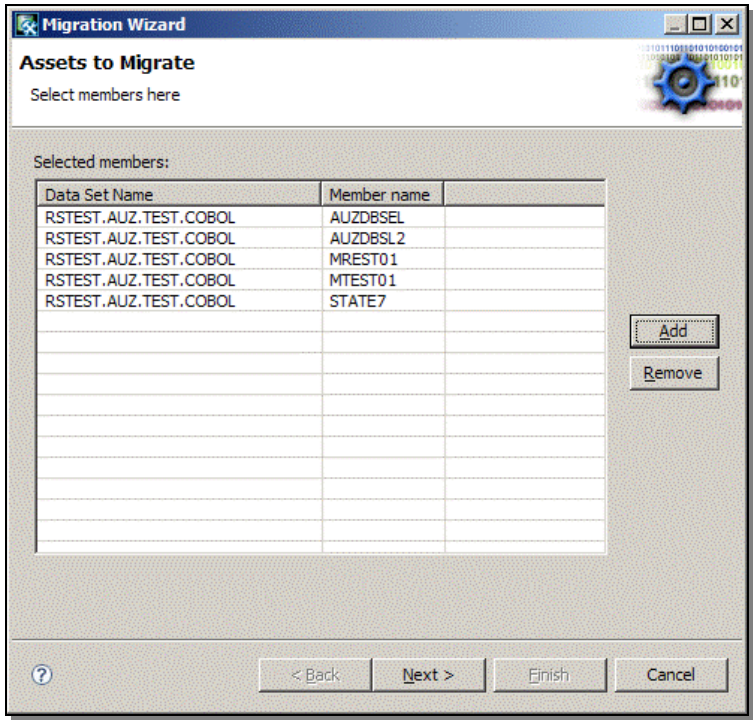

Figure 42: The selected assets to migrate are listed within the Migration Wizard.

Click **Next** when you are ready to continue.

Within the window titled Migrate Asset Options, specify the group into which to migrate the assets. Remember that assets can only be migrated into the lowest level group within the project hierarchy. In this case study, groups DEV1 and DEV2 are the lowest level groups within the project hierarchy, so only those two groups will appear in the selection list when you click the drop-down arrow.

Next, specify which data set type to migrate the assets into. For example, if the data set you migrated assets from was a COBOL data set, you would then migrate the assets into a similar data set. Use the drop-down arrow within the Type field to select a data set from the list of types defined to this project. In this case study, the assets will be stored in the project's COBOL data set.

The language field specifies which language definition should be used on these objects when building the assets. Use the drop-down arrow within the Language field to select a language definition to use with all assets.

Note: *Because only one language definition can be used when migrating assets en masse, you must migrate assets requiring different language definitions separately.*

The default authorization code is used. The assets will be conditionally migrated, which means that if there is a problem during the migration process, the migration will stop. The SCLM default date and time stamp will be used for these assets. Click **Next** when you are ready to migrate the assets.

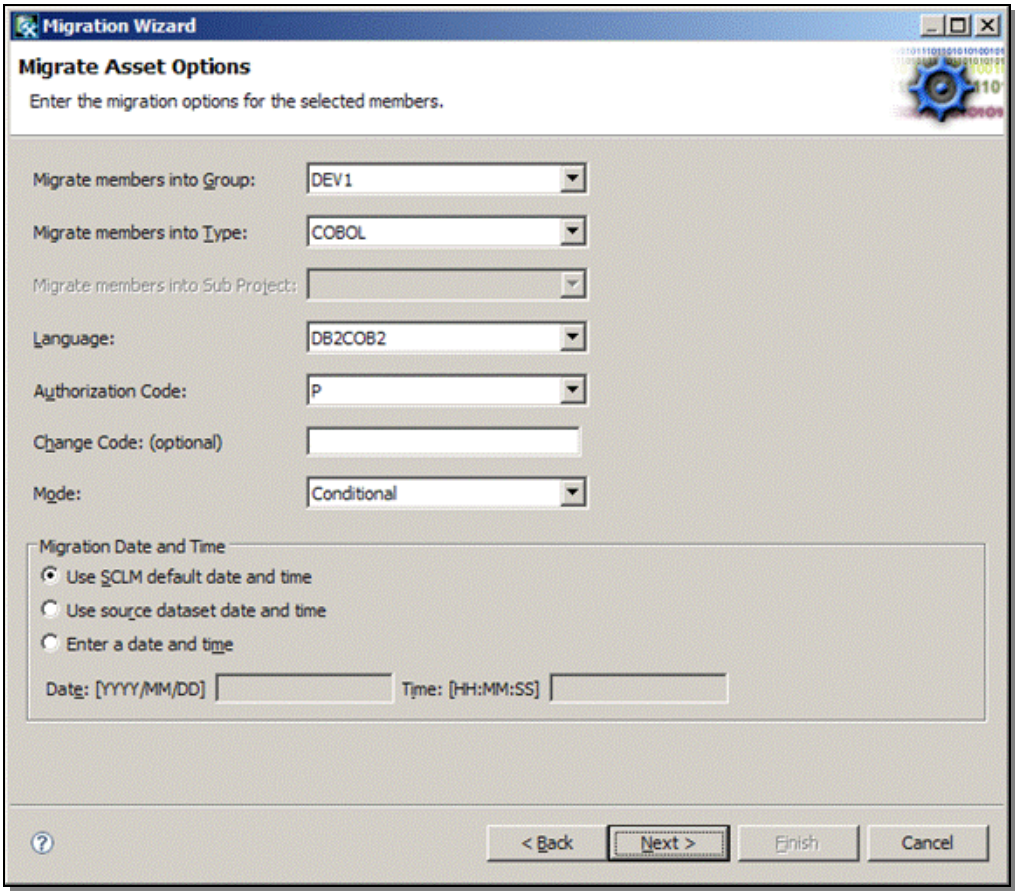

Figure 43: Specify the options to use to migrate the assets.

SCLM migration services are called, and the Administrator Toolkit registers each asset within the accounting and auditing data sets.

When the migration is complete, a report is returned for your review, as shown in Figure 44. A return code is issued for each asset that was migrated into the project. Click **Finish** to complete the migration process and exit the Migration Wizard.

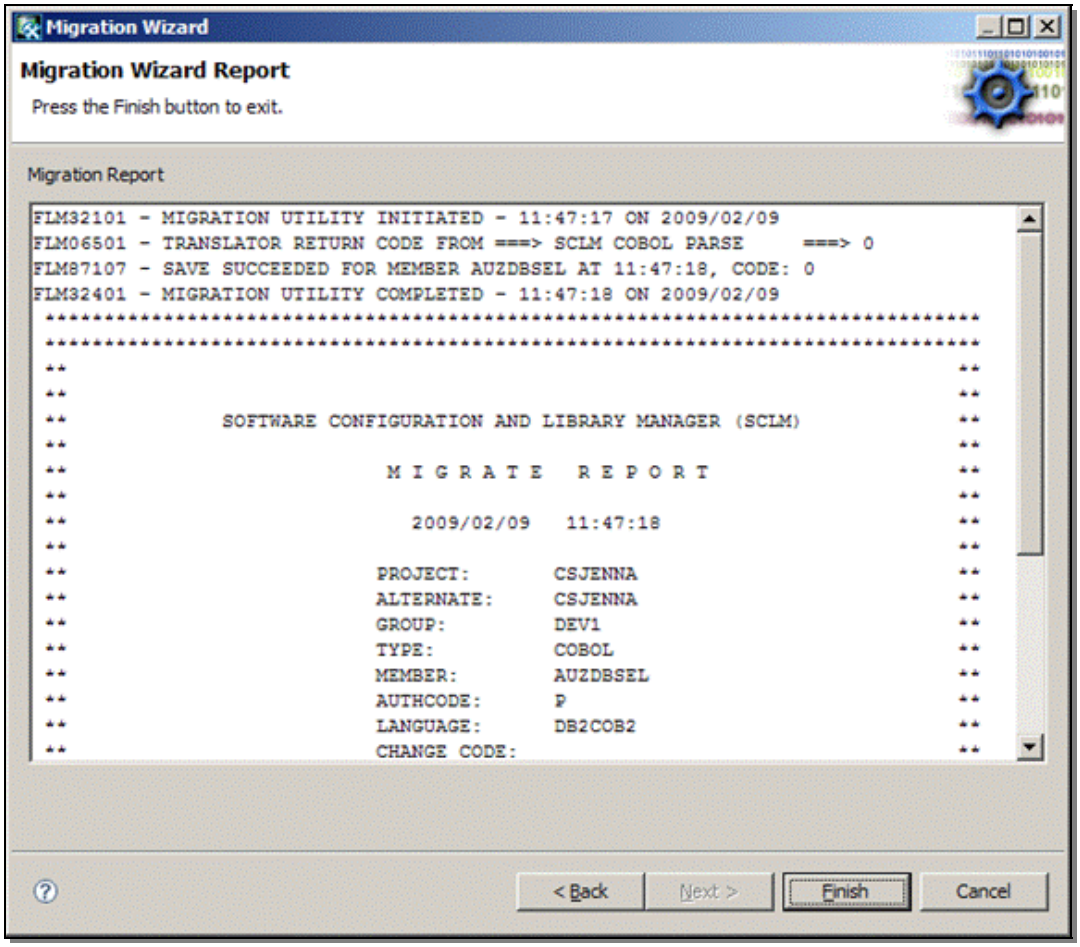

Figure 44: The Migration Report is returned for your review.

Once your project definition is complete, you have built an executable load module, and have optionally migrated any assets into your project, you must then build architecture definitions.

3.11 Create Architecture Definition

Architecture Definitions tell SCLM how pieces of a project's source code fit together and are crucial to the structure of a project. You can automatically generate SCLM architecture definitions from an existing load module or parse existing JCL into an architecture definition. A manual option for creating architecture definitions from scratch also exists.

For this case study, multiple architecture definitions will be created for the new DB2 project. A few different kinds of architecture definitions are required for a DB2 project. You will need something called a BINDDEF architecture definition to complete the DB2 binds, a CCDEF architecture definition to control the compilation of your source code, and a LECDEF

architecture definition that controls the link-edit process. There is also a high-level architecture definition that contains all subsequent DB2 architecture definitions. The high-level architecture definition is built within SCLM, which in turn causes all included architecture definitions to get built, and BINDS to be run, resulting in an executable DB2 program. Remember that a 'build' is the process in which raw source code is assembled or compiled and link-edited to create an executable program.

The Administrator Toolkit can automate this entire process through the Architecture Definition Wizard, assisting you in creating the specific architecture definitions required for your project. Using content-sensitive fields and drop-down boxes, you can easily create any kind of architecture definition with less mistakes, resulting in a usable architecture definition.

To invoke the Architecture Definition Wizard, right-click the project on which the architecture definitions are to be created. When the pop-up menu is displayed, select the option titled Architecture Definitions, as shown below in Figure 45.

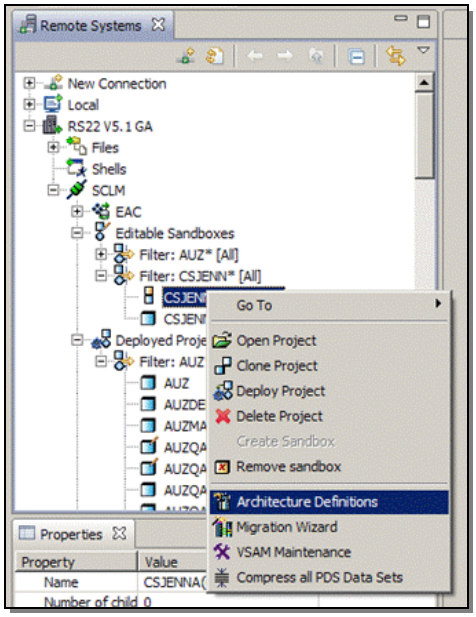

Figure 45: Invoke the Architecture Definition Wizard.

When the window titled Architecture Definition is displayed, you must first select which type of architecture definition to create from the bottom left-hand frame titled Create new Architecture Definition, as shown below in Figure 46 on page 41. In this case study, the radio button titled DB2 Architecture Definition is selected.

By default, the ArchDef Group and ArchDef Type fields are defaulted, but their values may be changed by clicking on the drop-down arrow within each field and selecting a new value. However, when you are creating architecture definitions for a DB2 project, and you have selected the radio button titled Create new Architecture Definition, then only the field titled ArchDef Group may be changed. This is because the Architecture Definition Wizard will automatically present you with the appropriate windows that will be used to create the different types of architecture definitions required for a DB2 project, and it ignores the field titled ArchDef Type.

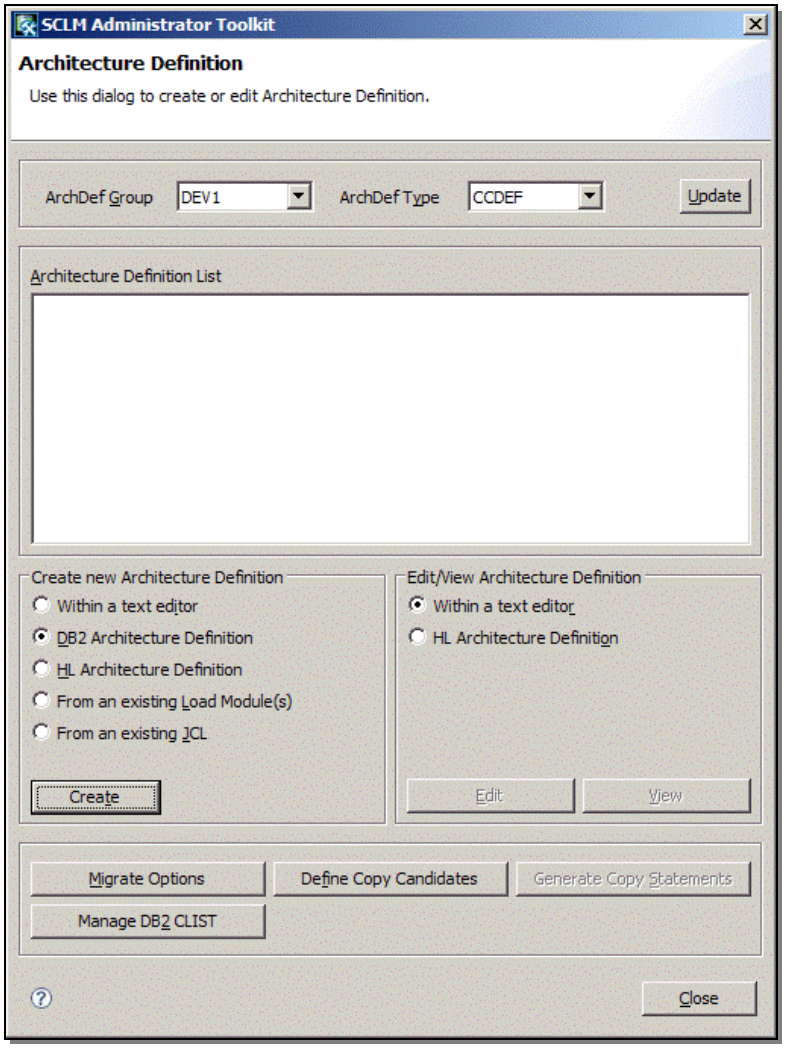

Figure 46: Select which kind of Architecture Definition to create.

Click the button titled **Create** when you are ready to create DB2 architecture definitions.

A window titled Create DB2 Architecture Definition will appear, as shown below in Figure 47, allowing you to specify the name of each type of architecture definition that will be created, and the data set type in which to store each kind of architecture definition.

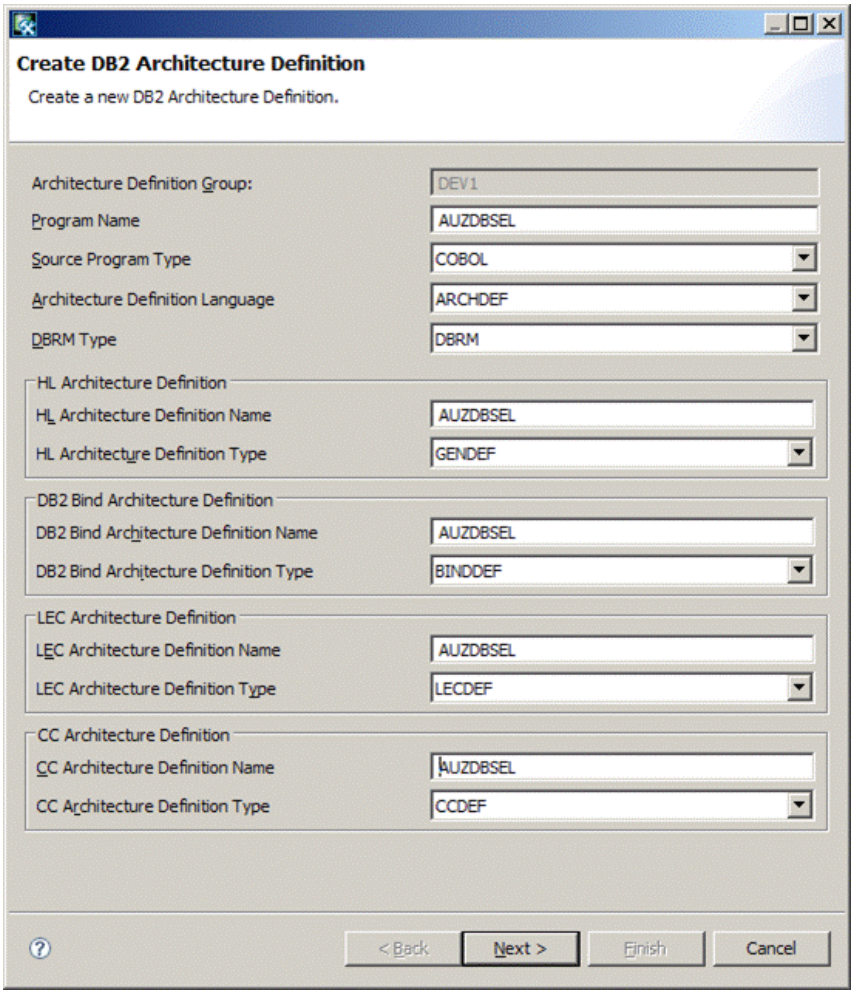

Figure 47: Specify the name and data set type for each kind of architecture definition.

Provide the following required information within this window to create each kind of architecture definition for your DB2 project.

- **Architecture Definition Group:** This field displays the group you specified on the previous window for which to create the architecture definition and cannot be edited.
- **Program Name:** The DB2 program for which the architecture definition is being created.
- **Source Program Type:** The data set type where the DB2 program can be located.
- **Architecture Definition Language:** The language to use to create this architecture definition.
- **DBRM Type:** The data set type where the DB2 DBRM should be placed.

- **HL Architecture Definition:** This section lists the name and type of the high level architecture definition. The high level architecture definition refers to the lower-level architecture definitions.
  - HL Architecture Definition Name: Provide up to an 8-character name to call the new high level architecture definition.
  - HL Architecture Definition Type: Specify the data set type where the high level architecture definition should be stored.

- **DB2 Bind Architecture Definition:** The BIND architecture definition tells SCLM how and where to run DB2 BIND commands for the DB2 program referenced in the field Source Program Name.
  - DB2 Bind Architecture Definition Name: Provide up to an 8-character name to call the new DB2 BIND architecture definition.
  - DB2 Bind Architecture Definition Type: Specify the data set type where the DB2 BIND architecture definition should be stored.

- **LEC Architecture Definition:** The link-edit control architecture definition tells SCLM where to put the output from running a link-edit, what language to use when running the link-edit and where to store the listing.
  - LEC Architecture Definition Name: Provide up to an 8-character name to call the new link-edit control architecture definition.
  - LEC Architecture Definition Type: Specify the data set type where the link-edit control architecture definition should be stored.

- **CC Architecture Definition:** The Compilation Control architecture definition tells SCLM where to put the object module and where it can find the DB2 program.
  - CC Architecture Definition Name: Provide up to an 8-character name to call the new compilation control architecture definition
  - CC Architecture Definition Type: Specify the data set type where the compilation control architecture definition should be stored.

When you have completed specifying the DB2 architecture definition information, click **Next** to continue.

The first kind of architecture definition that will be created is the CCDEF architecture definition. Remember that this kind of architecture definition controls the compilation of the source code for the specified DB2 program. Some values will already be filled in for you based on values you previously defined, as shown in Figure 48. Use the drop-down arrows within the empty fields to specify blank values.

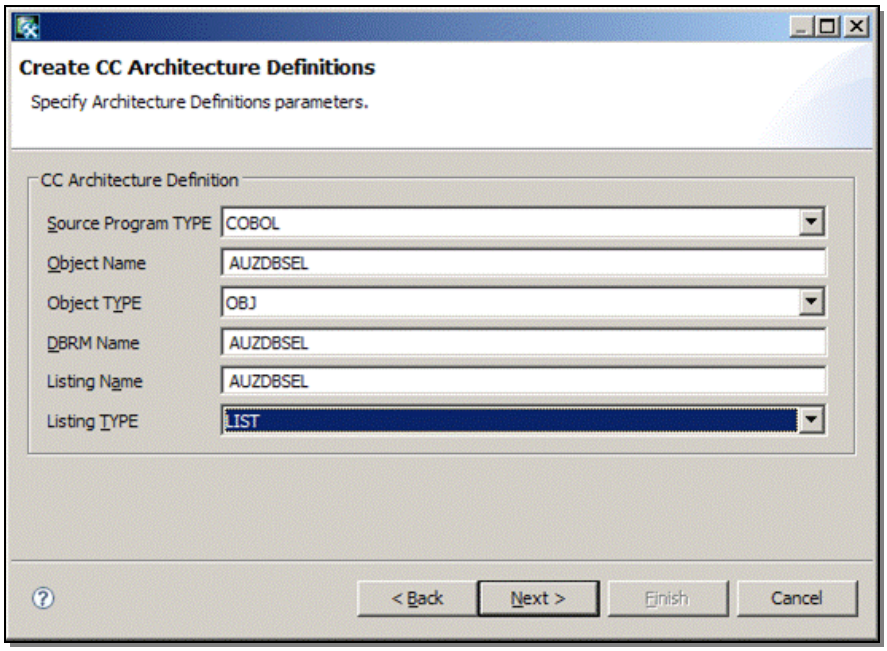

Figure 48: Create the Compilation Control architecture definition.

Specify the values for the fields Object Name and Listing Type. Descriptions for the fields on this window follow:

- **Source Program Type:** This value should already be filled in for you as it was specified on a previous window. This is the data set type where the DB2 program can be located.
- **Object Name:** This is the name of the object module to be created when SCLM builds the DB2 architecture definition. Its value is defaulted to the same name as the Source Program name provided on the previous window.
- **Object Type Name:** This is defaulted to the data set type where the object module is stored.
- **DBRM Name:** This value was defaulted to the same name as the Source Program Name provided on a previous window. The DBRM module is given this value and stored in the DBRM data set type.
- **Listing Name:** This value was defaulted to the same name as the Source Program Name. It is the member name of the listing that gets created when SCLM builds the compilation control architecture definition.
- **Listing Type:** Specify the data set type where you want the listing stored. The listing is created as a result of SCLM building the compilation control architecture definition.

When you have completed filling in the required values on this panel, click **Next**.

The second kind of architecture definition to be created is the LECDEF architecture definition. Remember that this kind of architecture definition controls how source code is link-edited. Some values will already be filled in for you based on values you previously defined, as shown in Figure 49. Use the drop-down arrows within the empty fields to specify blank values.

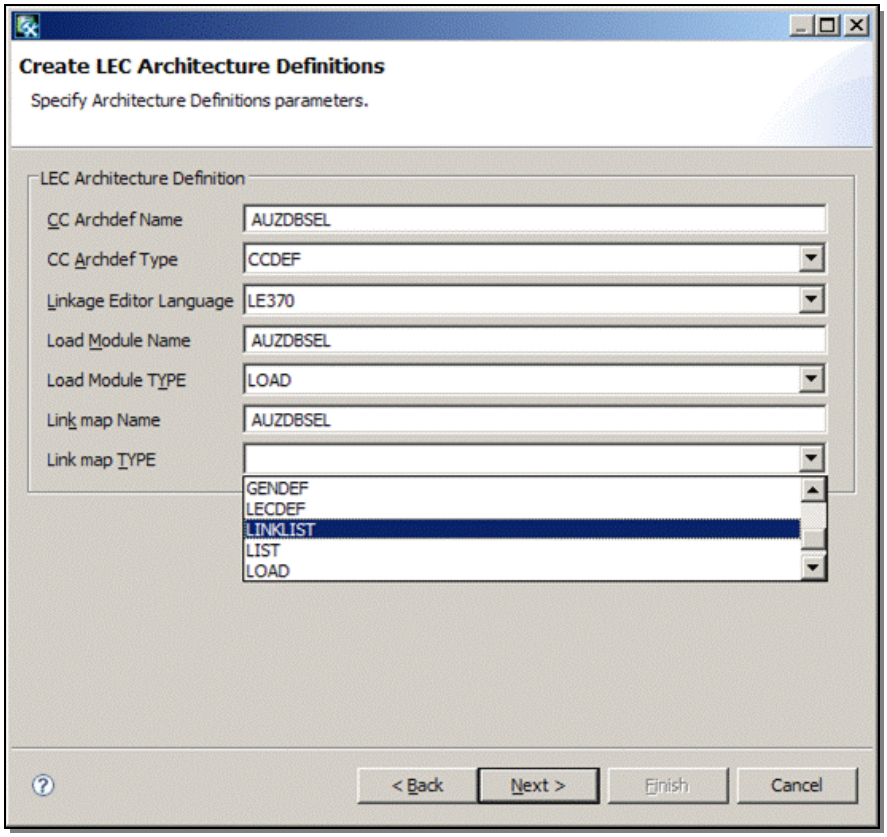

Figure 49: Create the Link Edit Control architecture definition.

Specify values for the fields Load Module Type, and Link Map Type. Descriptions for the fields on this window follow:

- **CC Archdef Name:** The name of the compilation control architecture definition you specified on the initial DB2 architecture definition panel.
- **CC Archdef Type:** This is the data set type where the CC architecture definition is located, and was specified on a previous panel.
- **Linkage Editor Language:** This is the language to be used to process the link-edit control architecture definition and was specified on a previous panel.
- **Load Module Name:** The load module name is defaulted to the same name as the Source Program Name specified on a previous panel. This is the load module to be created as a result of building the link-edit control architecture definition.
- **Load Module Type:** Specify the data set type where the load module is to be stored when it is created.

- **Link Map Name:** A link map gets generated as a result of building the LEC architecture definition. By default this value is the same name as the Source Program Name specified on a previous panel.

- **Link Map Type:** Specify the data set type where the link map is to be stored when it is created.

When you have completed filling in the required values on this panel, click **Next**.

The third kind of architecture definition is the BINDDEF, which is the DB2 architecture definition that is used when executing the BINDs for your DB2 program. Some values will already be filled in for you based on values you previously defined, as shown in Figure 50. Use the drop-down arrows within the empty fields to specify blank values.

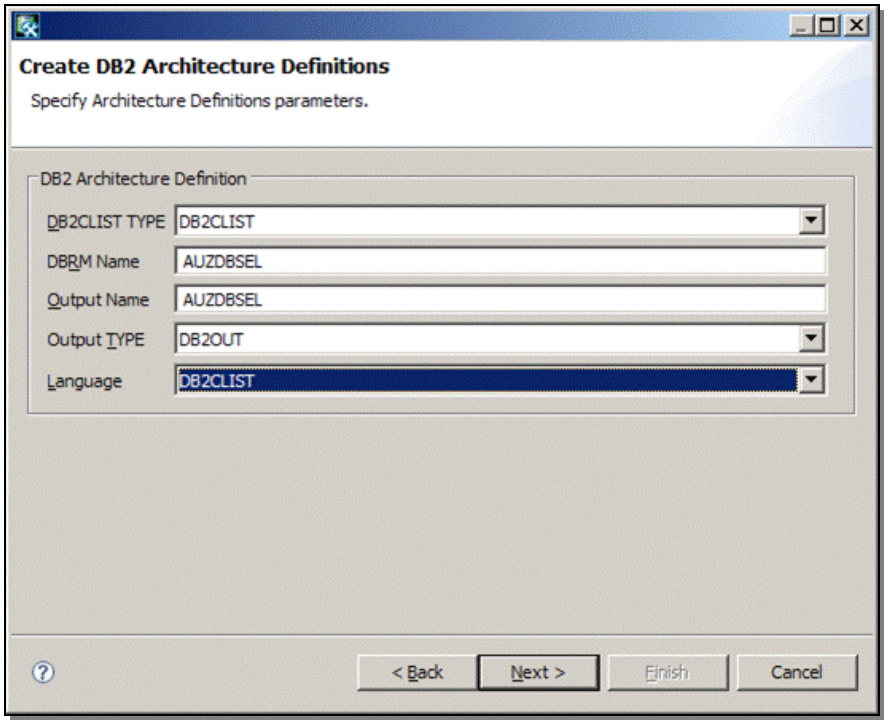

Figure 50: Create the DB2 BIND architecture definition.

Specify values for the fields Output Type and Language. Descriptions for the fields on this window follow:

- **DB2CLIST Type:** The data set type where the DB2CLIST is generated for the BIND commands to use when the DB2 BIND architecture definition is built within SCLM. By default, the value is DB2CLIST.

- **DBRM Name:** This value is the name of the DBRM that gets created as a result of building the DB2 architecture definitions in SCLM and was defaulted to the same value as the Source Program Name specified on a previous window.

- **Output Name:** The name of the member to be created and stored in the data set type referenced in Output type. By default it is the same value as the Source Program Name provided on a previous window.

- **Output Type:** Specify the data set type where the output of the DB2 bind architecture definition is stored.
- **Language:** Specify the language to be used to run the DB2 BIND architecture definition.

When you have completed filling in the required values on this panel, click **Next**. A window titled Create HL Architecture Definitions is displayed listing those values you supplied to create the CCDEF, the BINDDEF and the LECDEF, as shown in Figure 51. Remember that a High Level architecture definition simply references those architecture definitions required to create an executable DB2 program. It is this architecture definition that you execute a Build against to create load modules.

The values on this window are filled in with the values you specified when creating each kind of architecture definition, and should not be changed. If you must change one of the architecture definitions' values, click the button titled **Back** to navigate to the specific architecture definition whose values you want to change.

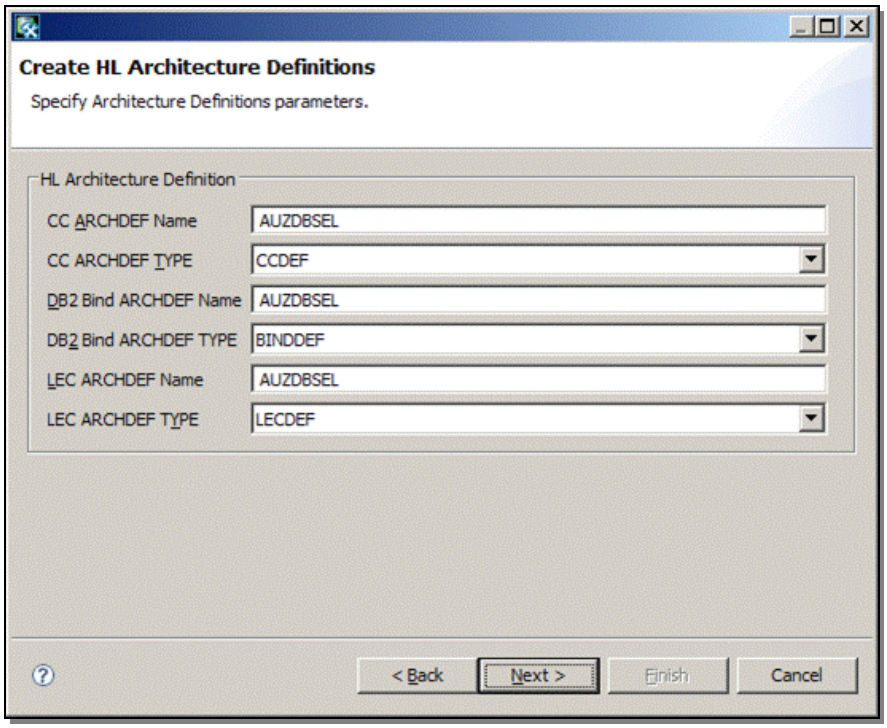

Figure 51: Create the High Level architecture definition.

Click **Next** to continue. Each kind of architecture definition is presented to you in an editor window in the order in which they were created.

The Compilation Control architecture definition is displayed first, as shown below in Figure 52.

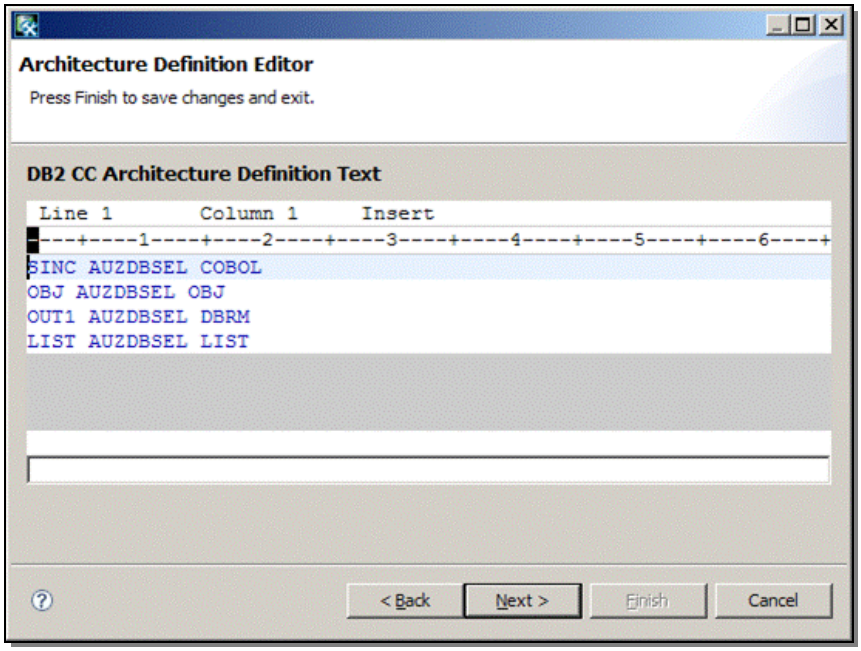

Figure 52: The compilation control architecture definition is returned for your review.

Click **Next** through each architecture definition to continue. The Link-Edit Control Architecture Definition is displayed next, as shown in Figure 53.

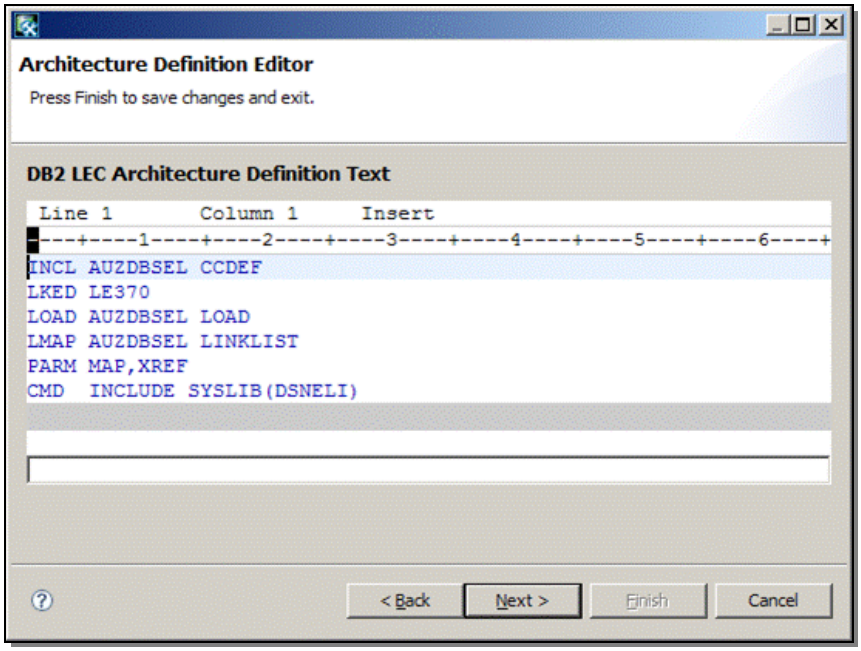

Figure 53: The link edit control architecture definition is returned for your review.

Click **Next** through each architecture definition to continue.

The Bind Architecture Definition is displayed next, as shown in Figure 54. This architecture definition executes the DB2 CLIST (displayed on the next page) which binds the program packages when the high level architecture definition is built in SCLM.

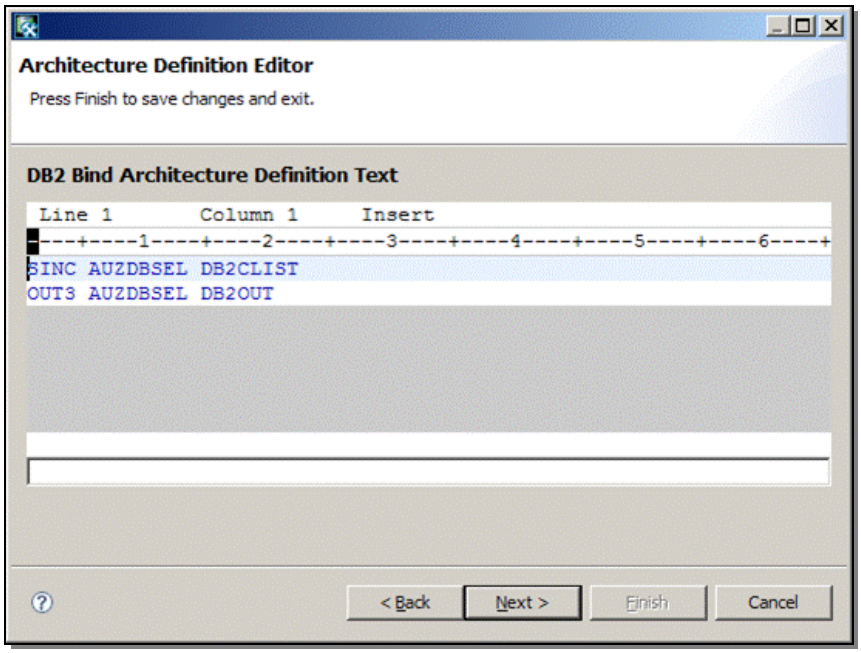

Figure 54: The DB2 bind architecture definition is returned for your review.

Click **Next** through each architecture definition to continue.

The DB2 CLIST is displayed for your review and should not be edited. The DB2 CLIST is run to execute the binds when the DB2 Bind Architecture Definition (BINDDEF) is built in SCLM. The syntax statement %INCLUDE references the DBRM name specified within the BINDDEF, as shown in Figure 55.

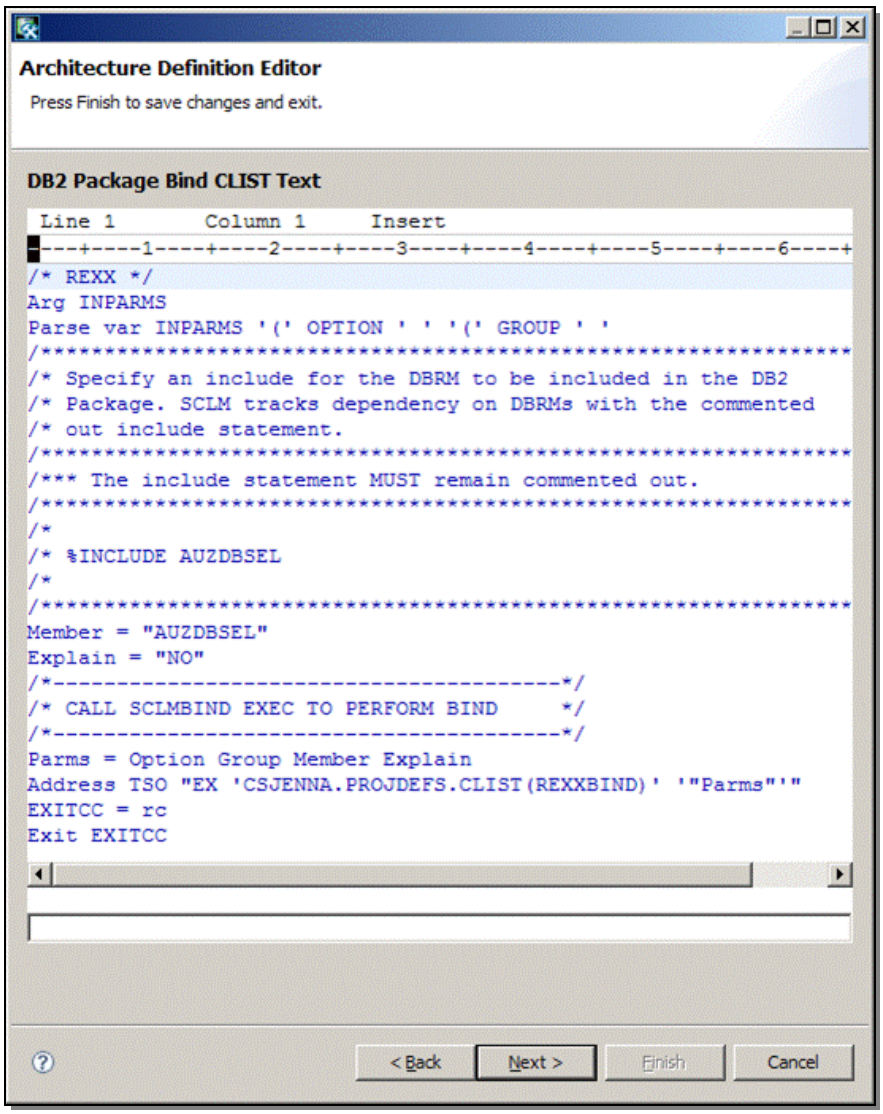

Figure 55: The DB2 CLIST executes the binds when the DB2 Bind Architecture Definition is built in SCLM.

Click **Next** through each architecture definition to continue.

The high level architecture definition is displayed last, as shown in Figure 56. It simply references all lower-level architecture definitions, to include the CCDEF, the BINDDEF, and the LECDEF. When building a DB2 project's assets in SCLM, a build is done against the high level architecture definition, causing all assets referenced within the lower level architecture definitions to be built. Remember that the process of building an asset in SCLM takes raw source code and turns it into executable programs.

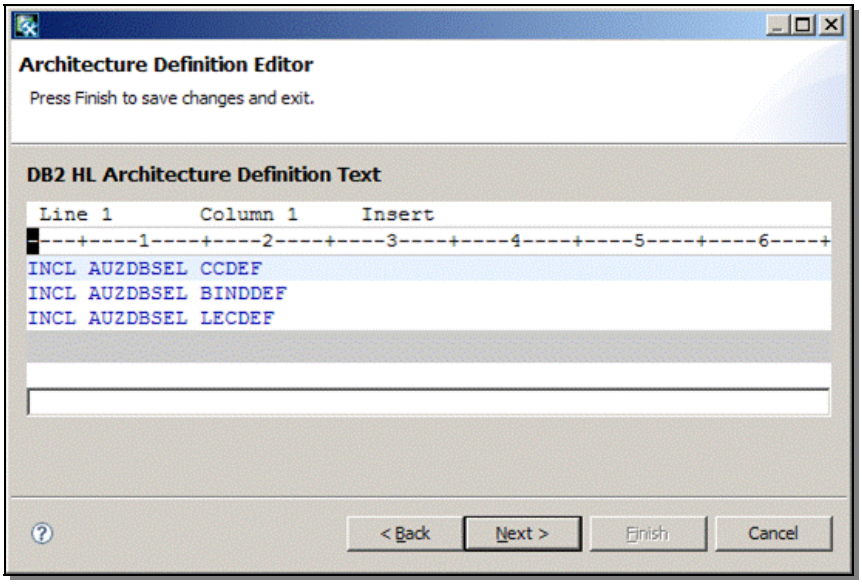

Figure 56: The High Level architecture definition is returned for your review.

Click **Next** to continue.

When the window titled Architecture Definition Migration Options is displayed, as shown in Figure 57, use the drop-down arrows where appropriate to specify values for the following parameters:

- **Architecture Definition Language:** Specify the language with which to create the architecture definitions. This language will be used when building the high level architecture definition. This parameter is required. There is no default.
- **Authorization Code:** Specify the authorization code defined in the project to use when building the assets named in this architecture definition. This parameter is required.
- **Change Code:** Specify up to an 8 character change code to use to indicate the reason for an update or modification to a member controlled by SCLM. This parameter is optional.
- **Mode:** Specify how SCLM is to proceed if an error is encountered. This parameter is required. The default is Conditional, which will stop the create process if there are any warnings or errors.

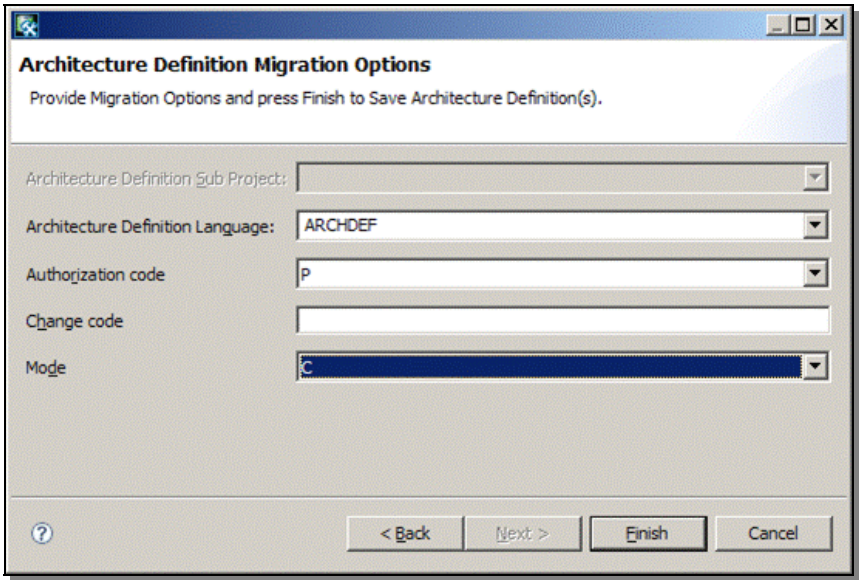

Figure 57: Specify the parameters with which to create the architecture definitions.

Click Finish to create the architecture definitions. A message will be returned when the architecture definitions have been created, as shown in Figure 58.

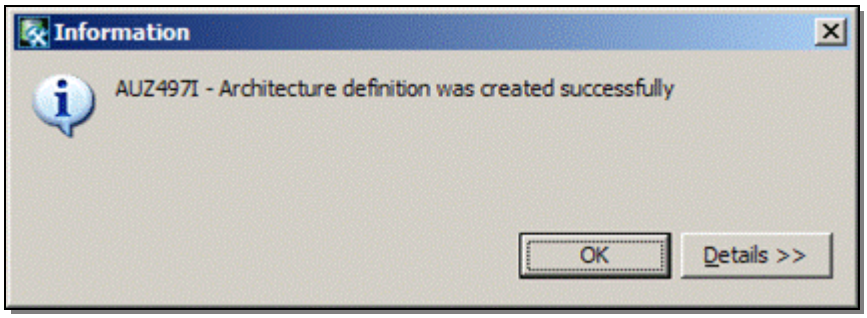

Figure 58: Informational messages are returned about the create process.

Clicking on the button titled Details will result in expanding the Information window to display additional information about the message. Click **OK** to close the informational message window and return to the Architecture Definition Wizard.

Troubleshooting: *Should an error be returned during any task, the task is written to the Reports node within the Remote Systems view, as discussed in this document in the subsection titled* A Brief Introduction*. It is these reports that Product Support will request when opening a PMR with IBM for diagnosis.*

You may view and optionally edit any architecture definitions that have been created for your project by using the Architecture Definition Wizard. On the window titled Architecture Definition, specify the ArchDef Group and ArchDef Type that contains the architecture definition you want to view. Click **Update** to display all architecture definitions within the group/Type data set.

In this example, the group/Type data set DEV1.CCDEF is displayed, as shown in Figure 59. Since there is only one within that group/Type data set, only one appears within the window.

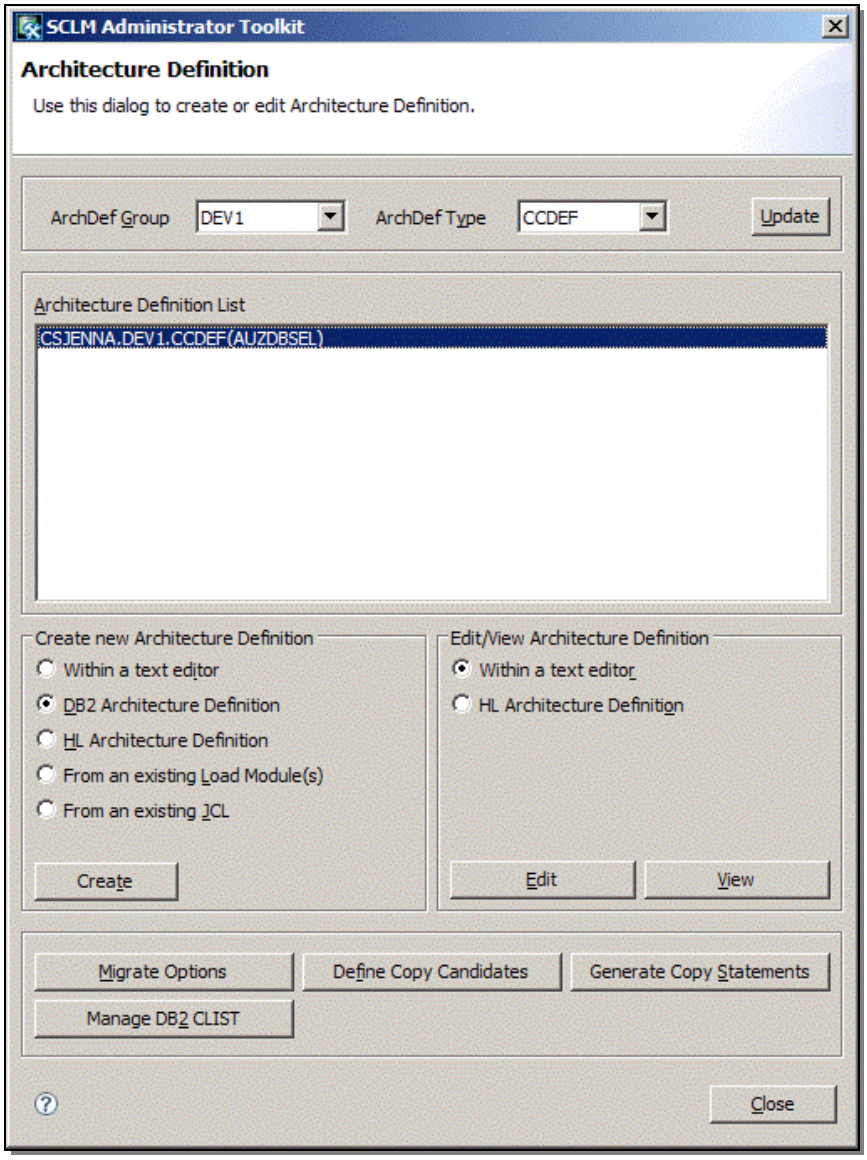

Figure 59: View or edit architecture definitions.

Highlight the architecture definition within the list, and either click the button titled Edit to make changes to the definition, or click the button titled View to simply view the definition.

Click the button titled **Close** to exit the Architecture Definition Wizard.

3.12 Test the project

Once you have created architecture definitions within the Administrator Toolkit, you can then test your project within SCLM to ensure the project definition you have defined will function correctly before deploying the project into production. To test your project, you can simply build

and promote the high level architecture definition that was created in the previous section. This will produce a load module that can be executed as a program.

To build the high level architecture definition for the new project, invoke SCLM, then provide the project name and development group, as shown in Figure 60. Select option **3** to invoke the Utilities panel. Press **Enter** to continue.

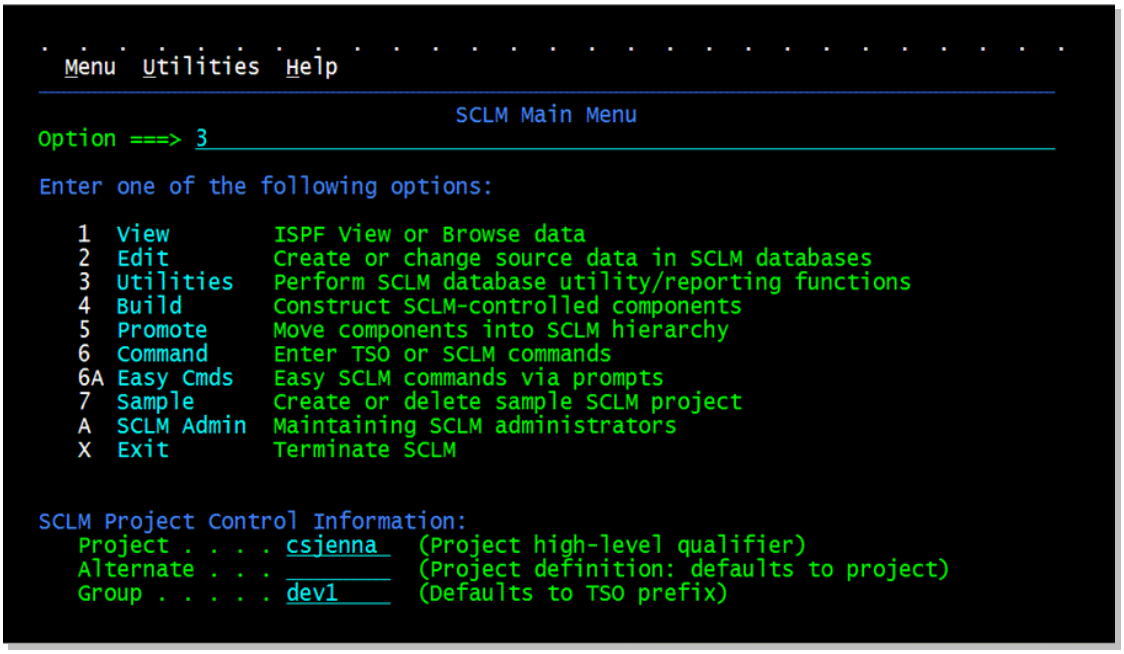

Figure 60: Specify the SCLM project whose assets you want to access.

On the panel titled SCLM Utilities Menu, select option **1** to access project assets, as shown in Figure 61. Press **Enter**.

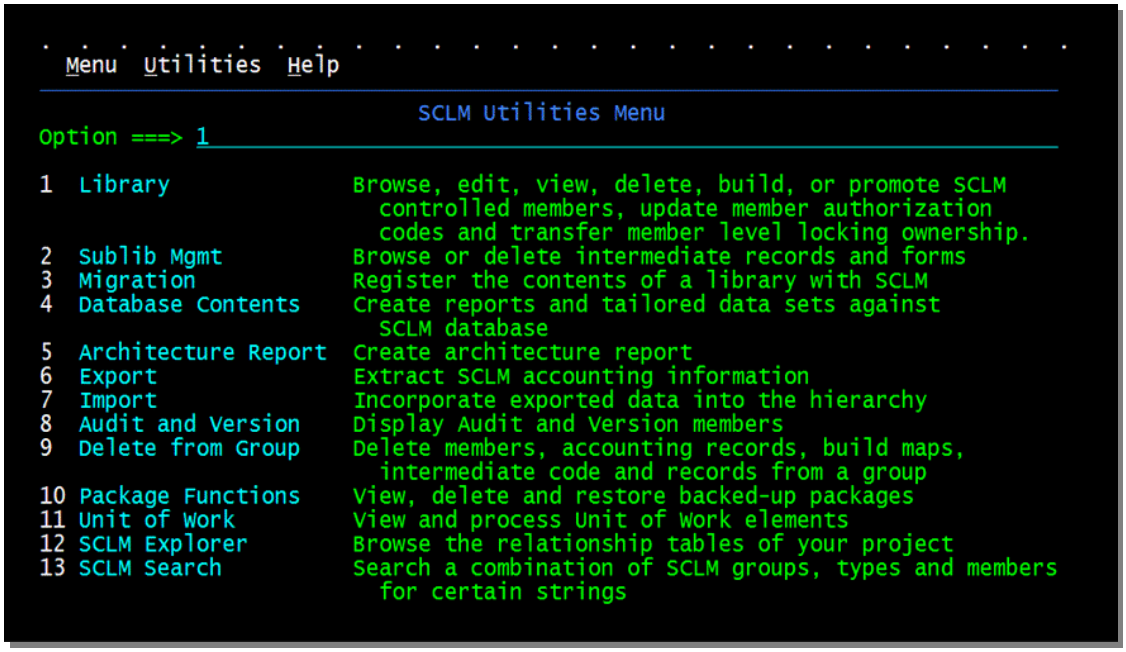

Figure 61: Select option 1 to view the contents of a library.

On the panel titled SCLM Library - Entry Panel, specify the data set type where the high level architecture is located. In this case study, the data set type which contains the high level architecture definition is GENDEF, as shown in Figure 62. You may leave the field titled Member blank to receive a list of PDS members belonging to the group/Type data set specified. Ensure the option titled Hierarchy View is selected on the bottom of the screen. Press **Enter**.

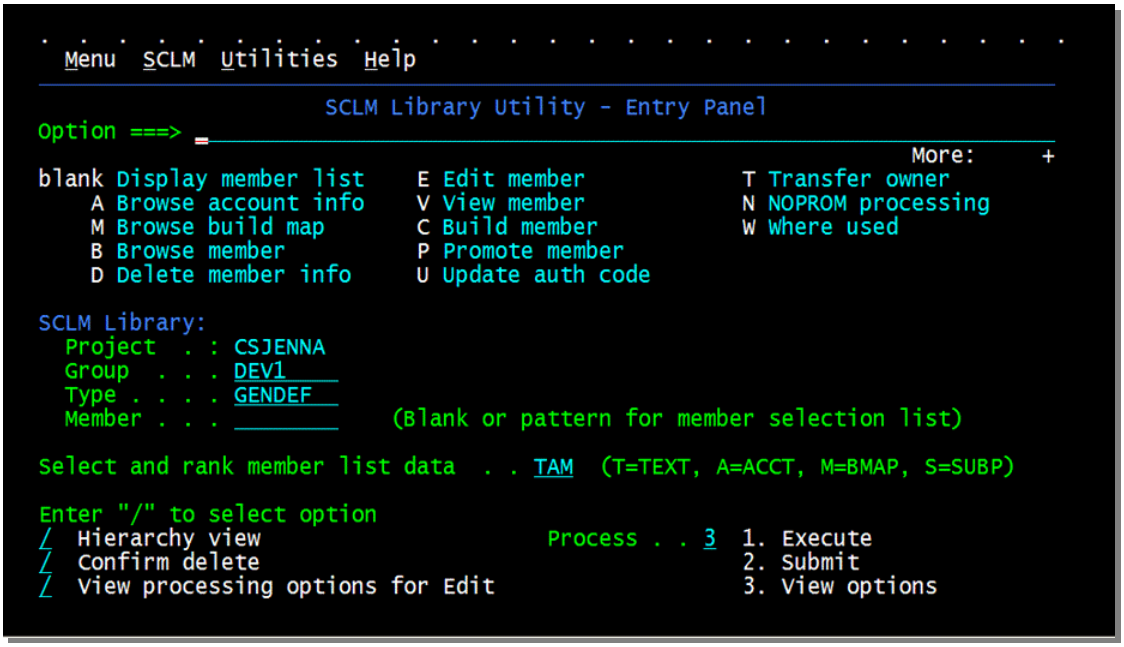

Figure 62: Specify the data set type where the architecture definition is stored.

On the panel titled Hierarchy View, type a C next to the high level architecture definition to Build the architecture definition, as shown in Figure 63. Remember that building the high level architecture definition will build the low-level architecture definitions and the associated source code. Press **Enter**.

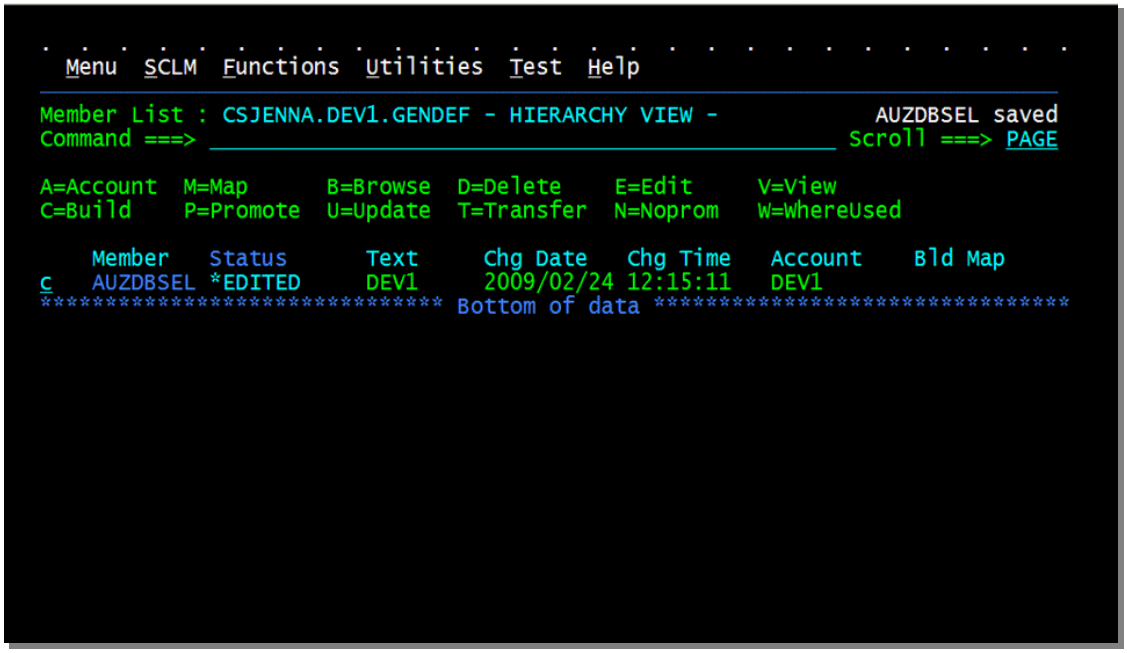

Figure 63: Build the architecture definition by selecting option C.

On the panel titled SCLM Build – Entry Panel, ensure the information within the fields Project, Group, Type and Member are correct before pressing Enter, to ensure the specified asset is the one you want to build, as shown in Figure 64. Press **Enter**.

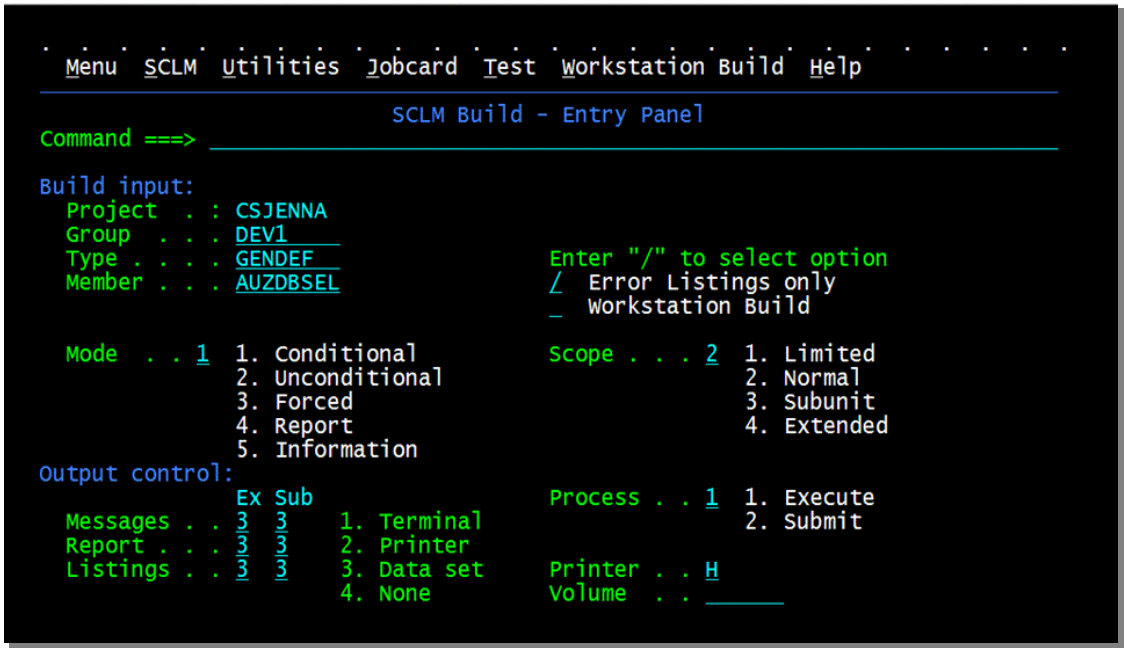

Figure 64: Double check the asset specified is the one you want to build.

When the SCLM Build function is invoked for the high level architecture definition all assets referenced within the architecture definition is assembled and linked, or compiled to create output that is either an executable program or a module that is then used as input to another executable program. That is, building an architecture definition causes the assets defined within the architecture definition to build as well.

The following Build Report is generate by the SCLM Build function and can be written to a data set that can either be kept or deleted, or the Build Report can be displayed on the screen and deleted upon exiting the Build Report. The options used during the Build function are listed within the comment section at the top of the Build report.

Example 1: SCLM Build report from high level architecture definition AUZDBSEL.

```
********************************* Top of Data ********************************
******************************************************************************
******************************************************************************
**                                                                          **
**                                                                          **
**             SOFTWARE CONFIGURATION AND LIBRARY MANAGER (SCLM)             **
**                                                                          **
**                          B U I L D    R E P O R T                       **
**                                                                          **
**                          2009/02/24   12:28:55                          **
**                                                                          **
**                           PROJECT:      CSJENNA                         **
**                           GROUP:        DEV1                            **
**                           TYPE:         GENDEF                          **
**                           MEMBER:       AUZDBSEL                        **
**                           ALTERNATE:    CSJENNA                         **
**                           SCOPE:        NORMAL                          **
**                           MODE:         CONDITIONAL                     **
**                                                                          **
**                                                                          **
******************************************************************************
******************************************************************************
    ******* B U I L D   O U T P U T S   G E N E R A T E D *******  Page 1

MEMBER     TYPE       VERSION      KEYWORD
------     ----       -------      -------
AUZDBSEL   OBJ              2      OBJ
AUZDBSEL   LIST             2      LIST
AUZDBSEL   DBRM             2      OUTX
AUZDBSEL   DB2OUT           1
AUZDBSEL   LOAD             1      LOAD
AUZDBSEL   LINKLIST         1      LMAP
         ******* B U I L D   M A P S   G E N E R A T E D *******  Page 2

                                  (REASON FOR REBUILD)
MEMBER     TYPE       VERSION      MEMBER     TYPE
------     ----       -------      -------    ----
AUZDBSEL   BINDDEF          1      ********************
AUZDBSEL   CCDEF            2      AUZDBSEL   COBOL
AUZDBSEL   GENDEF           1      ********************
AUZDBSEL   LECDEF           1      ********************
    ******* B U I L D   O U T P U T S     D E L E T E D   *******  Page 3

MEMBER     TYPE       VERSION      KEYWORD
------     ----       -------      -------
        ******* NO MODULES DELETED *******
         ******* B U I L D   M A P S     D E L E T E D   *******  Page 4

                                  (REASON FOR DELETE)
```

```
MEMBER     TYPE       VERSION      MEMBER     TYPE
------     ----       -------      -------    ----
         ***** NO BUILD MAPS DELETED *****
******************************** Bottom of Data ********************************
```

Once your asset has been successfully built, it must be promoted up through the project hierarchy. Promoting an asset means that you are moving the asset and its dependent objects up the hierarchy into the next highest group. Testing the promote process after a build checks to ensure the project's Group definitions are correct. When a promotion occurs within SCLM, accounting information is updated for all assets being promoted so that SCLM can track the lifecycle of that asset.

If an error occurs within the Build process, possible causes might be an error made within the source code itself, incorrect language definition parameters or incorrect parameters defined within the associated architecture definition. Review the SCLM messages to determine what the likely cause of the build failure is.

To promote the high level architecture definition for the new project, select option P on the panel titled Hierarchy View, as shown in Figure 65. Remember that promoting the high level architecture definition will build the low-level architecture definitions and the associated source code. Press **Enter**.

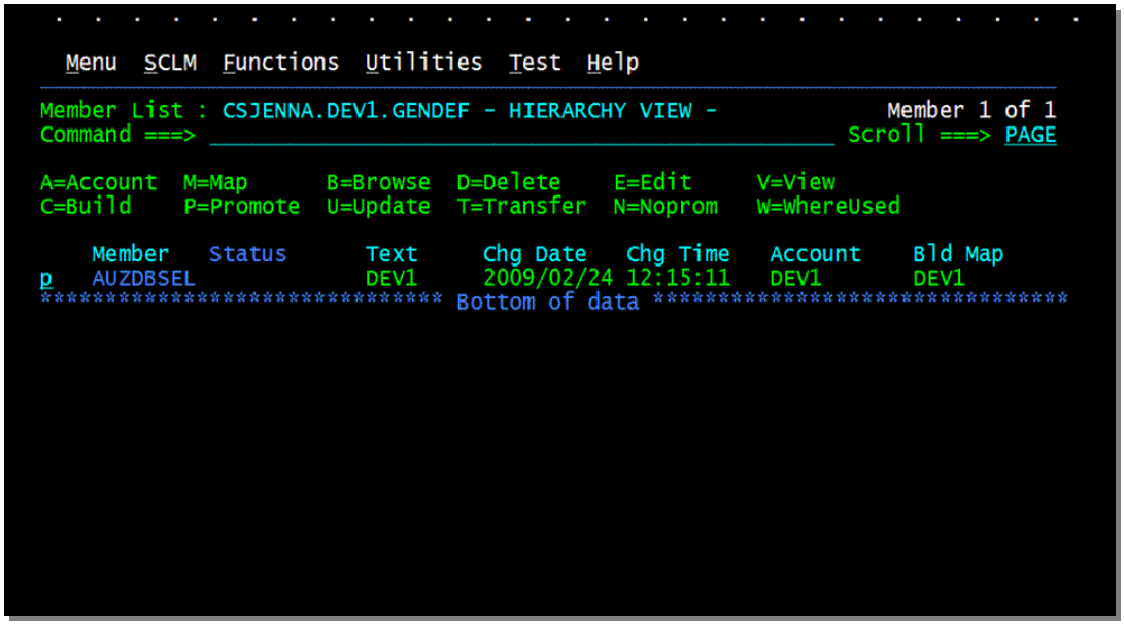

Figure 65: Promote the architecture definition by selecting option P.

On the panel titled SCLM Promote – Entry Panel, ensure the information within the fields Project, Group, Type and Member are correct before pressing Enter, to ensure the specified asset is the one you want to promote, as shown in Figure 66. Press **Enter**.

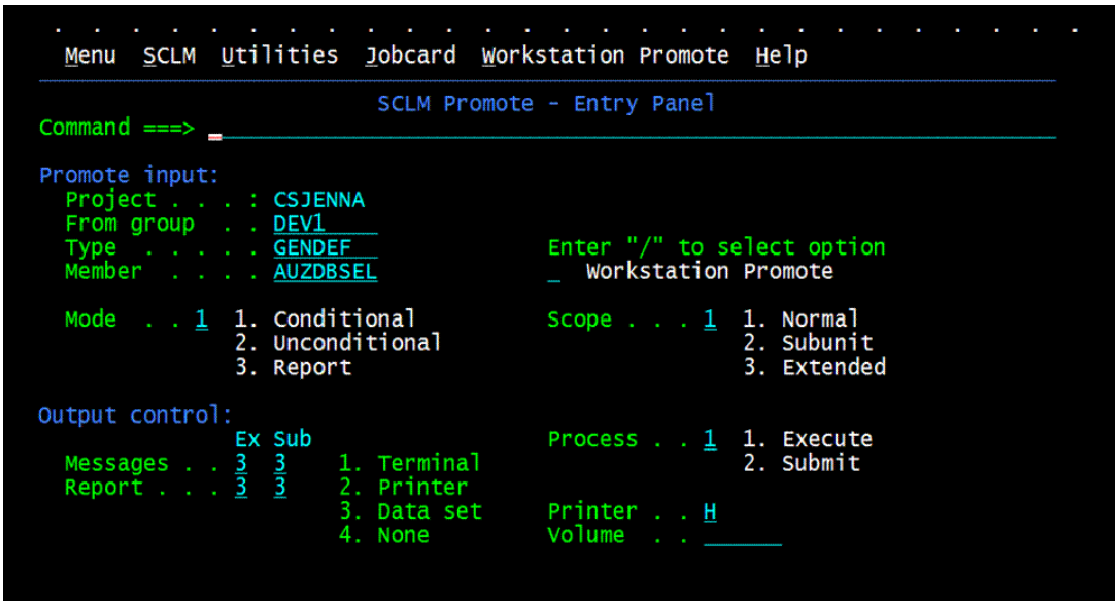

Figure 66: Double check the asset specified is the one you want to promote.

When the SCLM Promote function is invoked for the high level architecture definition, all assets referenced within the architecture definition are promoted as well. Accounting and auditing information for the assets being promoted are updated by the SCLM Promote function.

The following Promote Report is generated by the SCLM Promote function and can be written to a data set that can either be kept or deleted, or the Promote Report can be displayed on the screen and deleted upon exiting the Promote Report. The options used during the Promote function are listed within the comment section at the top of the report.

Example 2: SCLM Promote report from high level architecture definition AUZDBSEL.

```
********************************* Top of Data ********************************
******************************************************************************
******************************************************************************
**                                                                          **
**                                                                          **
**            SOFTWARE CONFIGURATION AND LIBRARY MANAGER (SCLM)              **
**                                                                          **
**                      P R O M O T E    R E P O R T                        **
**                                                                          **
**                          2009/02/24   12:31:12                           **
**                                                                          **
**                                                                          **
**                          PROJECT:       CSJENNA                         **
**                          TO GROUP:      TEST                            **
**                          FROM GROUP:    DEV1                            **
**                          TYPE:          GENDEF                          **
**                          ARCH. MEM.:    AUZDBSEL                        **
**                          ALTERNATE:     CSJENNA                         **
**                          SCOPE:         NORMAL                          **
**                          MODE:          CONDITIONAL                     **
**                                                                          **
```

Example 2 Continued: SCLM Promote report from high level architecture definition AUZDBSEL.

```
**                                                                              **
** NOTE: "*" INDICATES "OUT OF SCOPE" ITEMS.                                    **
**********************************************************************************
**********************************************************************************
TYPE: BINDDEF
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
________  __________  ________  ________  __________  ____________

AUZDBSEL  2009/02/09  12:02:51                  X           X


TYPE: CCDEF
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
________  __________  ________  ________  __________  ____________

AUZDBSEL  2009/02/09  12:02:42                  X           X


TYPE: COBOL
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
________  __________  ________  ________  __________  ____________

AUZDBSEL  2009/02/24  12:28:37                  X           X


TYPE: DBRM
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
________  __________  ________  ________  __________  ____________

AUZDBSEL  2009/02/24  12:28:57                  X           X


TYPE: DB2CLIST
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
________  __________  ________  ________  __________  ____________

AUZDBSEL  2009/02/09  12:02:56                  X           X


TYPE: DB2OUT
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
________  __________  ________  ________  __________  ____________

AUZDBSEL  2009/02/24  12:28:59                  X           X


 TYPE: GENDEF
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
________  __________  ________  ________  __________  ____________

AUZDBSEL  2009/02/24  12:15:11                  X           X


 TYPE: LECDEF
```

Example 2 Continued: SCLM Promote report from high level architecture definition AUZDBSEL.

```
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
_________   __________  _______  _________          _________  ____________

AUZDBSEL    2009/02/09  12:02:47                       X          X


 TYPE: LINKLIST
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
_________   __________  _______  _________          _________  ____________
AUZDBSEL    2009/02/24  12:29:01                       X          X


 TYPE: LIST
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
_________   __________  _______  _________          _________  ____________

AUZDBSEL    2009/02/24  12:28:58                       X          X


 TYPE: LOAD
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
_________   __________  _______  _________          _________  ____________
AUZDBSEL    2009/02/24  12:29:00                       X          X


 TYPE: OBJ
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
_________   __________  _______  _________          _________  ____________

AUZDBSEL    2009/02/24  12:28:57                       X          X

**************************************************************************
**                                                                      **
**                        B U I L D    M A P S                        **
**                                                                      **
**************************************************************************
TYPE: BINDDEF
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
_________   __________  _______  _________          _________  ____________

AUZDBSEL    2009/02/24  12:28:55                       X          X


TYPE: CCDEF
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
_________   __________  _______  _________          _________  ____________
AUZDBSEL    2009/02/24  12:28:55                       X          X


TYPE: GENDEF
                                                    COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE            TEST       DEV1
_________   __________  _______  _________          _________  ____________
```

Example 2 Continued: SCLM Promote report from high level architecture definition AUZDBSEL.

```
AUZDBSEL  2009/02/24  12:28:55                X           X

TYPE: LECDEF
                                              COPIED TO  PURGED FROM
MEMBER      DATE        TIME     MESSAGE      TEST       DEV1
________  __________  ________  ________  __________  ____________

AUZDBSEL  2009/02/24  12:28:55                X           X

******************************** Bottom of Data ****************************
```

Once your asset has been successfully promoted up through the project hierarchy, you can test the program by executing it.

If there are errors during the promote process, the mostly likely cause is an error within the Group parameters defined within the project. Check the SCLM messages to ensure which parameter is the cause.

Example 3 contains the results of executing the DB2 sample program AUZDBSEL.

Example 3: Results of executing sample program AUZDBSEL from the new project.

```
****************************** TOP OF DATA ***************************
Card number: 4929717212000001
Expiration date: 200701
Card type: VISA
Card holder: John Johnson
Card number: 5100046590000001
Expiration date: 200801
Card type: Mastercard
Card holder: John Johnson
****************************** BOTTOM OF DATA ************************
```

If the program executes correctly, this confirms that your project's definition is accurate, that the language definitions are correct, and that the architecture definitions were defined accurately.

3.13 Deploy the project

Once you have safely tested your new project within the sandbox and are satisfied that the project definition is correct, you can deploy your new project into production. Deploying a project simply means that the contents of a sandbox's data sets are copied to the main project's data sets, and the sandbox data sets are deleted.

Once a sandbox project has been deployed into production, the project name will no longer appear within the node Editable Sandboxes within the Remote Systems view. Instead, the deployed project will appear within the node Deployed Projects. A project filter matching the project's name must exist before you can see the project within the list of Deployed Projects.

This last task will describe how to deploy a project that was created within a sandbox, then create a project filter within the Deployed Projects node so that the new project can be displayed within the list of deployed projects.

To deploy a project, simply highlight the project name within the node Editable Sandboxes. Then right click the project name to see the pop-up menu. When the pop-up menu is displayed, as shown in Figure 67, select the option titled Deploy Project.

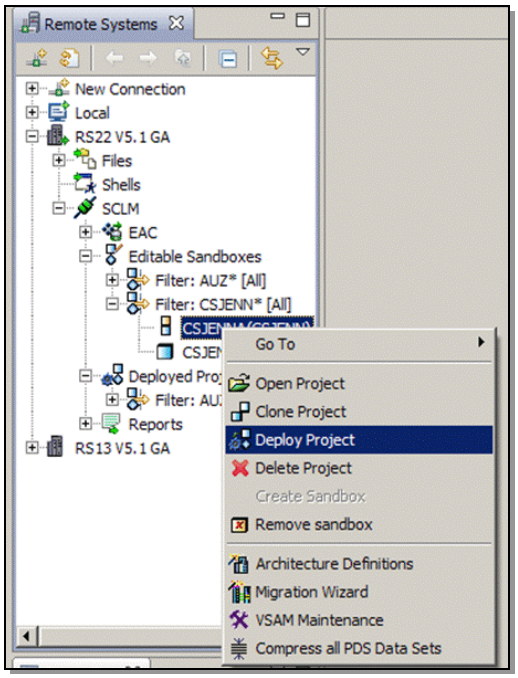

Figure 67: Select to deploy the sandbox project.

When the pop-up window titled Build Project Dialog is displayed, select the option titled Export after build. This will write the project definition into an external source data set. Click OK to continue.

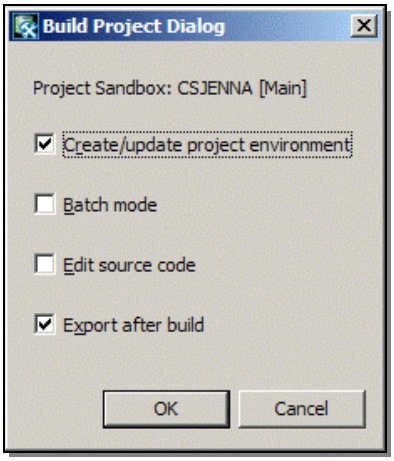

Figure 68: Export the new project's definition.

A message will be returned when the project has been deployed, as shown in Figure 69.

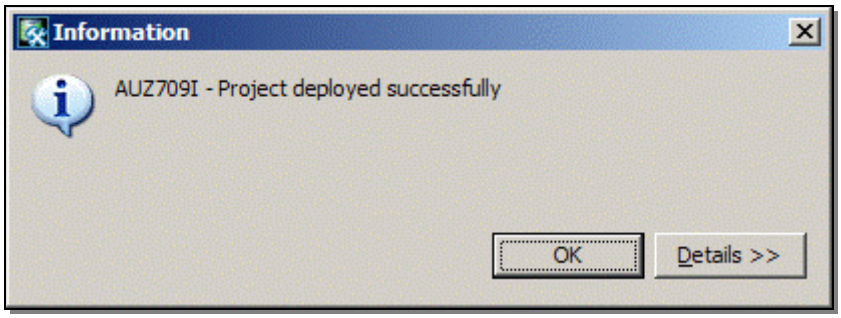

Figure 69: Informational messages are returned about the deployment.

Clicking on the button titled Details will result in expanding the Information window to display additional information about the message. Click **OK** to close the informational message window.

Troubleshooting: *Should an error be returned during any task, the task is written to the Reports node within the Remote Systems view, as discussed in this document in the subsection titled* A Brief Introduction*. It is these reports that Product Support will request when opening a PMR with IBM for diagnosis.*

A project filter whose filter criteria matches the new project name must exist before the new project can be viewed within the node Deployed Projects. If one does not exist, a new project filter can be added by highlighting the node Deployed Projects. When the pop-up menu is displayed, select the option New, then select the option Project Filter from the second pop-up menu, as shown below in Figure 70.

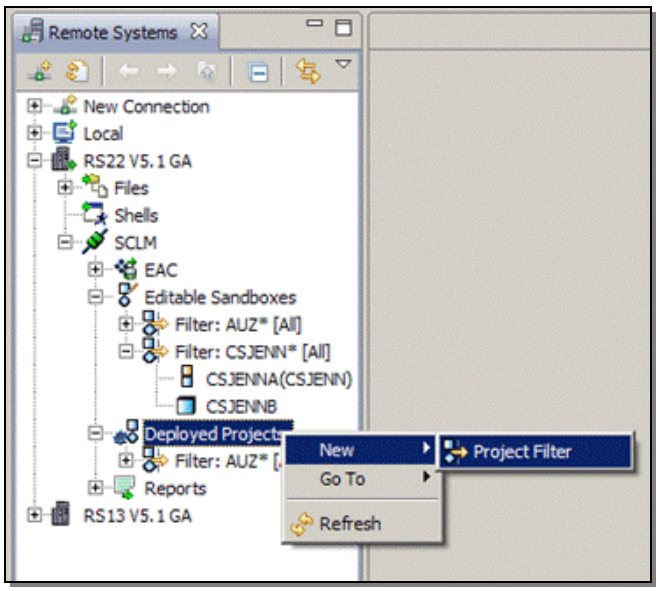

Figure 70: Create a new project filter.

When the window titled New SCLM Project Filter is displayed, provide a fully-qualified project name, or a partially qualified pattern using an asterisk as a wild card character, that matches the new project, as shown below in Figure 71.

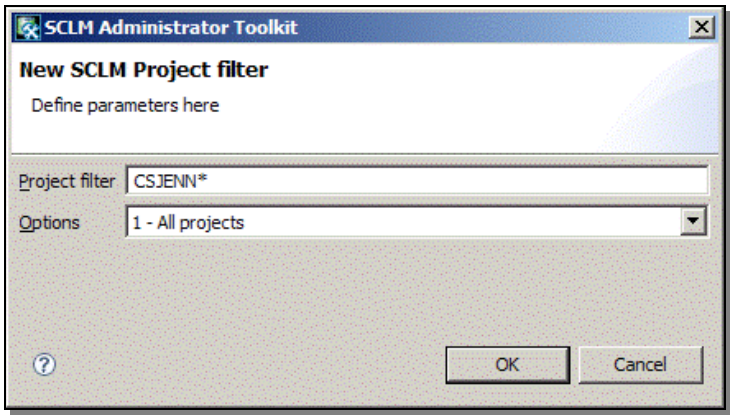

Figure 71: Provide the criteria for the project filter.

In this case study, the new project name was CSJENN (remember that the sandbox was called CSJENNA). To view a list of projects whose names match the specified criteria, an asterisk was specified as a wild card character. As a result, all projects whose name matches the specified criteria will be displayed when the project filter is expanded. Click **OK** when you have completed entering the project filter criteria.

The node Deployed Projects must be refreshed to pick up the new Project Filter that was just created. Right click the node Deployed Projects, and select the option titled Refresh from the pop-up menu, as shown in Figure 72.

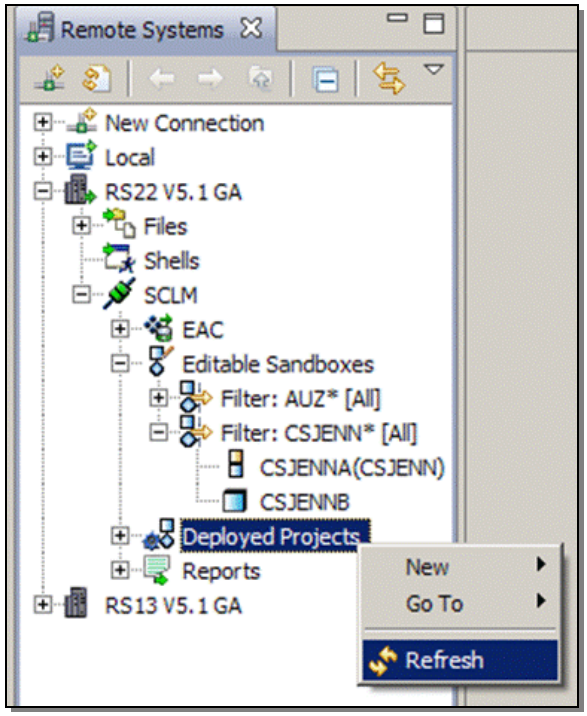

Figure 72: Refresh the Deployed Projects node to see the new project filter.

When the node Deployed Projects is refreshed, the project filters are displayed, as shown below in Figure 73. Click on the **plus sign** ('+') next to the project filter to expand the filter and view the newly deployed project.

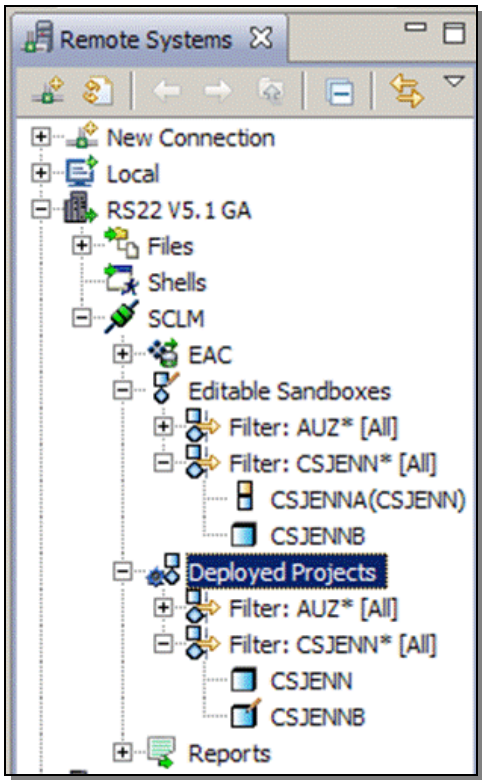

Figure 73: Collapse, then re-expand the node Deployed Projects to see the new project filter.

Once a project is deployed, it may still be edited should you decide changes need to be made. Simply double click the project within the Deployed Projects node. You will be asked if you want to create a sandbox for your project while changes are being made to the project. It is HIGHLY recommended that you create a sandbox to edit the project's definition. This will allow the project to remain intact while changes are being made to the sandbox, preventing costly errors from being introduced into the project currently being used within production.

When you select to create a sandbox for the deployed project you want to edit, you will see the sandbox project appear within a project filter in the node Editable Sandboxes. When you have completed making changes to the sandbox, you simply deploy the project following the instructions provided within this subsection of the whitepaper.

4 A Case for the Administrator Toolkit

Administering your project's definition isn't a daunting task when you use the Administrator Toolkit to assist you. Using the Administrator Toolkit to automate and streamline administration tasks, you can simplify complex tasks and reduce the amount of time it takes to complete them.

All SCLM macro parameters are available within the Administrator Toolkit, and main projects as well as alternate projects are fully supported.

Create sandboxes to test your new project in before making it available to your team members, or edit an existing project within a sandbox to ensure your changes give you the expected results without affecting the projects already in production.

Wizards within the Administrator Toolkit eliminate errors and help new users become accustomed to the nuances of SCLM, while seasoned administrators can automate processes that normally take time to complete.

The learning curve of creating and administering SCLM projects is reduced with a user-friendly interface in either ISPF panels or a workstation client. Using an intuitive flow, your project administrators will take less time performing administrative tasks, which will enable them to focus on other more business-critical tasks.

APPENDIX A: CREATE THE OBJECTS

The following DDL in Example 4 was used to create the DB2 objects used for this DB2 project. Simply copy it into your own dataset, then either run it through SPUFI or run it as a batch job using the sample JCL below in Example 7.

Example 4: Project DDL to create the necessary DB2 objects for this project.

```
******************************** Top of Data ***************************
create database dauzdb2;
commit;
create tablespace tauzdb2 in dauzdb2
segsize 4;
commit;
create table auzdb2.credit_card
       (card_num      varchar(16),
        exp_date      char(6),
        card_type     varchar(16),
        card_holder   varchar(40))
        in dauzdb2.tauzdb2;
commit;
create table auzdb2.transaction
       (card_num      varchar(16),
        amount        decimal (7,2))
        in dauzdb2.tauzdb2;
commit;
******************************** Bottom of Data ************************
```

The following DML in Example 5 was used to insert data into the 'CREDIT_CARD' table. This data is required for this DB2 project. Simply copy it into your own dataset, then either run it through SPUFI or run it as a batch job using the sample JCL below in Example 7.

Example 5: Project DML for the first table to insert the necessary DB2 data for this project.

```
******************************** Top of Data ************************
insert into auzdb2.credit_card
(card_num, exp_date, card_type, card_holder)
values ('4929717212000001',
        '200701',
        'VISA',
        'John Doe');
commit;
insert into auzdb2.credit_card
(card_num, exp_date, card_type, card_holder)
values ('5100046590000001',
        '200801',
        'Mastercard',
        'John Doe');
commit;
******************************** Bottom of Data ************************
```

The following DML in Example 6 was used to insert data into the 'TRANSACTION' table. This data is required for this DB2 project. Simply copy it into your own dataset, then either run it through SPUFI or run it as a batch job using the sample JCL below in Example 7.

Example 6: Project DML for the second table to insert the necessary DB2 data for this project.

```
******************************** Top of Data ***********************
insert into auzdb2.transaction
(card_num, amount)
values ('4929717212000001',
        +458.26);
commit;
insert into auzdb2.transaction
(card_num, amount)
values ('4929717212000001',
        -238.57);
commit;
insert into auzdb2.transaction
(card_num, amount)
values ('5100046590000001',
        +127.65);
commit;
insert into auzdb2.transaction
(card_num, amount)
values ('5100046590000001',
        -476.35);
commit;
******************************** Bottom of Data ********************
```

The following sample JCL in Example 7 can be used to create the DB2 objects and populate the tables with data. Before you can run this JCL, it must be modified to fit your shop's requirements.

Example 7: Sample JCL to create the DB2 objects and populate the data.

```
******************************** Top of Data ***********************
//#JOBCARD#
//*
//*
//**********************************
//*
//* DB2 SAMPLE SQL PROGRAM
//*
//**********************************
//*
//JOBLIB DD DSN=#DB2.SDSNLOAD#,DISP=SHR
//BATCHSQL EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(#SSID#)
RUN PROGRAM(DSNTEP2) -
LIB('#DB2.RUNLIB.LOAD#')
END
/*
//SYSIN DD DISP=SHR,DSN=#YOUR.SQL.DATASET#(#CREATEDDL#)
//      DD DISP=SHR,DSN=#YOUR.SQL.DATASET#(#INSERTDML#)
//      DD DISP=SHR,DSN=#YOUR.SQL.DATASET#(#INSERTDML#)
/*
******************************** Bottom of Data ********************
```

The following variables within the sample JCL must be changed before the batch job can be submitted.

#JOBCARD#

The proper job card information required for batch jobs within your IT department.

#DB2.SDSNLOAD#

The name of the DB2 load library that contains program IKJEFT01.

#SSID#

The DB2 subsystem on which the objects will be created.

#DB2.RUNLIB.LOAD#

The name of the DB2 library that contains program DSNTEP2.

#YOUR.SQL.DATASET#

The name of the data set in which the DDL to create the tables is stored.

#CREATEDDL#

The PDS member name in which the CREATE syntax is stored.

#INSERTDML#

The PDS members in which the INSERT syntax is stored for both the CREDIT_CARD table and the TRANSACTION table.

APPENDIX B: THE COBOL PROGRAM

The following program is migrated into the SCLM project in section 3.10 of this Whitepaper titled “Migrate Project Assets”. Copy this COBOL program into a PDS data set. It is from this PDS data set that the COBOL program will be migrated into your project.

Example 8: Sample COBOL program for the DB2 project.

```
************************** Top of Data ************************
 id division.
 program-id.
     auzdbsel.
*
*    Rocket Software SCLM Administrator Toolkit
*    Sample program to use for DB2 Language
*    Definitions and Architecture Definitions.
*
*    Copyright, 2007, Rocket Software, Inc.
*    All rights reserved.
*
*    This program displays the contents of the credit card
*    table.
*
*    Revision    Date    Author    Reason
*       00     04/23/07   CRMB     New program.
*
*    Note:
*    This program runs as a user batch or TSO program.
*
 data division.
 working-storage section.
 01  ws-dbesel-eye-catcher   pic x(80)
     value "auzdbsel".
 01  ws-sql-code             pic s9(9)  comp-5 value 0.
 01  not-found               pic s9(9)  comp-5 value +100.
     exec sql declare auzdb2.credit_card table
     (
         card_num varchar(16),
         exp_date char(6),
         card_type varchar(16),
         card_holder varchar(40)
     )
     end-exec.
     exec sql declare ccard-cursor cursor for
              select * from auzdb2.credit_card
     end-exec.
     exec sql include sqlca end-exec.
*    exec sql begin declare section end-exec.
*01  sqlcode                       pic s9(9)    comp-5.
*01  sqlstate                      pic x(5).
 01  ccard.
     02  card-num.
         49  card-num-len          pic s9(4)    comp-5.
         49  card-num-data         pic x(16).
     02  exp-date                  pic x(6).
```

```
     02  card-type.
         49  card-type-len          pic s9(4)    comp-5.
         49  card-type-data         pic x(16).
     02  card-holder.
         49  card-holder-len        pic s9(4)    comp-5.
         49  card-holder-data       pic x(40).
*    exec sql end declare section end-exec.
/
*
/
*
 procedure division.
 aa-main section.
 aa-0000.
*    exec sql whenever sqlerror goto aa-8000 end-exec
*    exec sql whenever sqlwarn  goto aa-8000 end-exec
     exec sql whenever not found continue    end-exec
*
     perform ba-process
     go to aa-9999
     .
 aa-8000.
     display "aa-sqlcode = " sqlcode
     .
 aa-9999.
     goback
     .
/    Main processing routine
*
 ba-process section.
 ba-0000.
     exec sql open ccard-cursor end-exec
     exec sql fetch ccard-cursor into :ccard end-exec
     if sqlcode not = not-found
         perform ca-print
     else
         display "ba-sqlcode = " sqlcode
     end-if
     exec sql close ccard-cursor end-exec
     .
 ba-9999.
     exit
     .
 ca-print section.
 ca-0000.
     perform until sqlcode not = 0
         display "Card number: "
                  card-num-data
         display "Expiration date: "
                  exp-date
         display "Card type: "
                  card-type-data
         display "Card holder: "
                  card-holder-data
         exec sql fetch ccard-cursor into :ccard end-exec
     end-perform
     .
 ca-9999.
     exit
     .
************************* Bottom of Data **********************
```

IBM United States of America

Produced in the United States of America