



Chicago Interface Group, Inc.

Greenhouse

*Concepts and
Facilities*

Chicago Interface Group, Inc.
858 West Armitage Avenue # 286
Chicago, IL 60614 USA

Phone: (773) 524-0998
Fax: (815) 550-6088
Internet: www.cigi.net
Email: support@cigi.net

*Copyright © 2008 Chicago Interface Group, Inc. All Rights Reserved
January 17, 2008*

*Greenhouse is a trademark of Chicago Interface Group, Inc.
Endevor is a registered trademark of Computer Associates International*

Table of Contents

Table of Contents	1-1
Chapter 1 : Concepts Overview	1-1
Introduction	1-3
Greenhouse Implementation	1-5
Summary	1-9
Chapter 2 : Facilities	2-1
Label Compliance Facility	2-3
Greenhouse Utilities	2-5
Additional Utilities	2-8
Lines of Code Reporting	2-9
Chapter 3 : ISPF Components	3-1
The Main Menu	3-3
Standard Build or Assign Function	3-6
Populating a Code Segment	3-7
Standard Export Function	3-8
Assign Function	3-9
Delta Sync Function	3-10
Standard Import Function	3-12
Standard Viewing Function	3-13
Labeling and Tagging Elements	3-14
Selection List Function	3-15
JCL Edit and Submit Function	3-16
Chapter 4 : Reports	4-1
GREENHOUSE Reports	4-2
GREENHOUSE Reports	4-3
Chapter 5 : Getting Started	5-1
Implementation Objectives for Clean Management	5-2
Decision Policy and Design Tables for Clean Management	5-3
Design Decisions for Clean Management	5-4
Implementation Objectives for Release Management	5-5
Decision Policy and Design Tables for Release Management	5-6
Design Decisions for Release Management	5-7

Implementation Objectives for Traditional and Non
Traditional Code Conversion Models..... 5-8
Design Decisions for Traditional Code Conversion
Implementations..... 5-9

Greenhouse Concepts and Facilities

Chapter 1 : Concepts
Overview

Introduction

Greenhouse provides both a foundation for and an immediate value to controlling your environment. In addition to being able to support massive levels of change, parallel development, and clean management, Greenhouse allows for the implementation of customer specific policies and change management configuration to support the movement of code and change.



There are many ways to implement Greenhouse. Its uses are as varied as the customers who install it. It is designed to be used for Clean Management, Parallel Management, and Release Management, and most customers use a combination of many functions and management disciplines.

Greenhouse for Clean Management

The general concept behind Greenhouse for Clean Management is that users want to control the access and update of code through the use of an external mechanism. They also want an audit report and tool to provide ongoing proof that renovated code is not being compromised. The Greenhouse mechanism for this is the Label Enforcement Rule Set, which allows customers to label components with a custom labeling scheme and then enforce the access of those components through enforcement exits. A good example of this would be preventing the retrieval of back levels of code.

Greenhouse for Release Management

The general concept of Greenhouse for Release Management is that many companies have multiple releases being developed simultaneously. These releases have overlapping code and dependencies that need to be tracked. Using the Greenhouse segment management facilities, users can be confident that they know what is contained in each release and where conflicts with other releases might occur. In addition, labeling provides a link back to source code levels in each segment.

Greenhouse for Parallel Management

The general concept of Greenhouse for Parallel Management is that users need to check out large segments of code for analysis and revisions. These segments of code will be sent to many PDSs, which will in turn either be exported to off-site repositories, shipped to a lab, or worked on locally. Regardless of where the work is being done and by whom, the fact remains that the code is outside of traditional change and configuration control. This reality is workable *only* if there is a mechanism that can collect and monitor the segments of work as a single entity, while the segments are out of the system and when they are brought back into the system.

Greenhouse Implementation

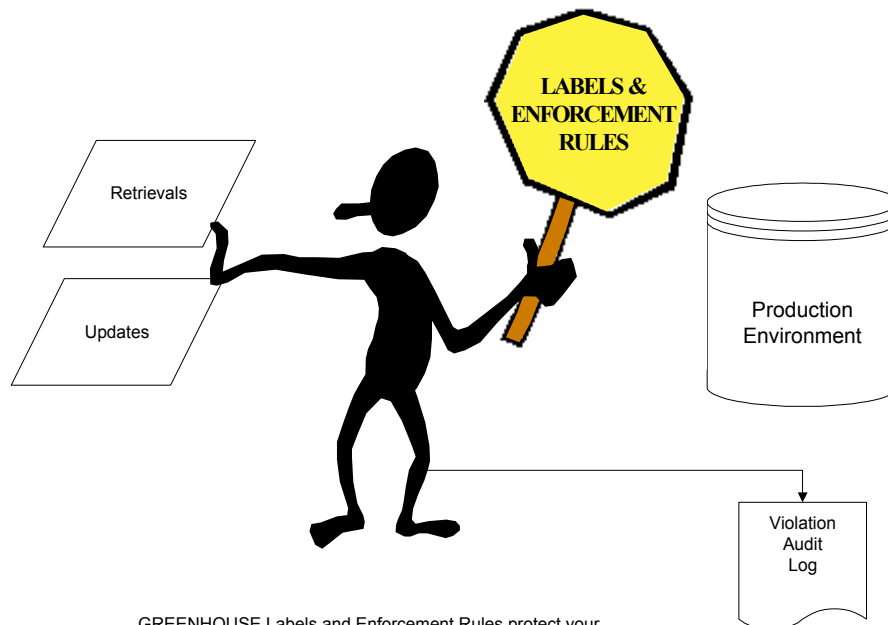
The versatility of the Greenhouse building blocks makes it possible for users to choose from several available methods of implementation, including:

- Clean Management
- Traditional Code Conversion Model
- Non-Traditional Code Conversion Model
- Release Management

Using Greenhouse for Clean Management

Clean Management is the process of controlling parallel development, certifying and labeling compliant code, tracking code certification, and protecting the code deemed as "ready" from non-certified changes. Clean Management requires a documented method and tool for tracking and reporting on certification, enforcement, testing, and exceptions.

Through the use of User Defined Labeling and Enforcement, Greenhouse users can quickly establish a repeatable and trackable process that captures activity against defined units of work.

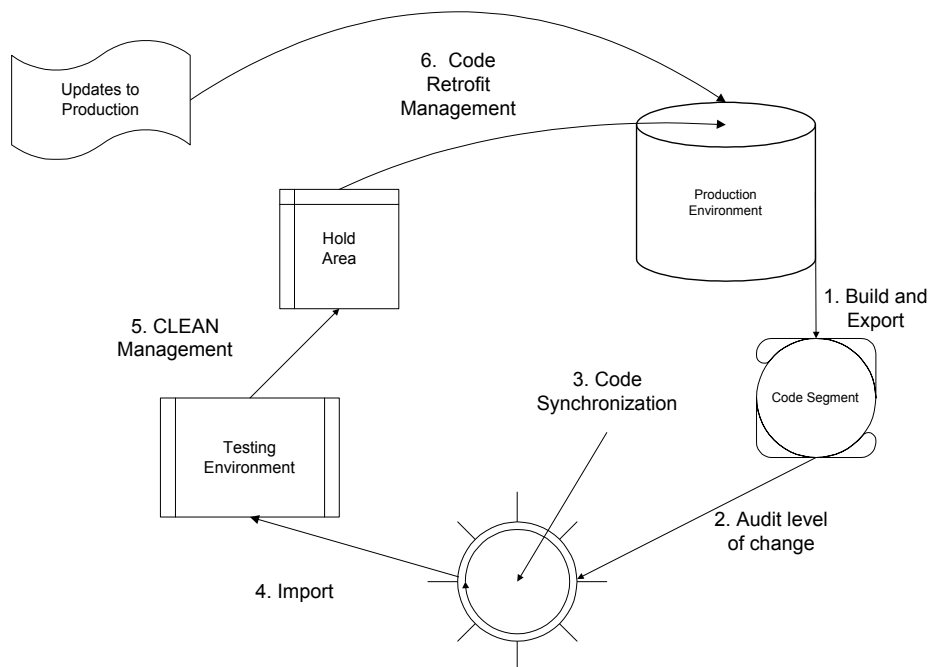


GREENHOUSE Labels and Enforcement Rules protect your production environment, and maintain an audit log of all rule violations.

Traditional Code Conversion Model

The Traditional Code Conversion model is comprised of several steps, each of which utilizes one or more separate product components:

Phase of Project	Related Product Components
Build and Export Code Segment	Build /Assign Function
Audit Level of Change	Lines of Code Reporting Delta Sync Utility
Code Synchronization	Delta Sync Code Merging Tool (CMT)
Import	Add-back Copy
Clean Management	Delta Sync Tagging Label Enforcement
Code Retrofit Management	Shifting Baseline

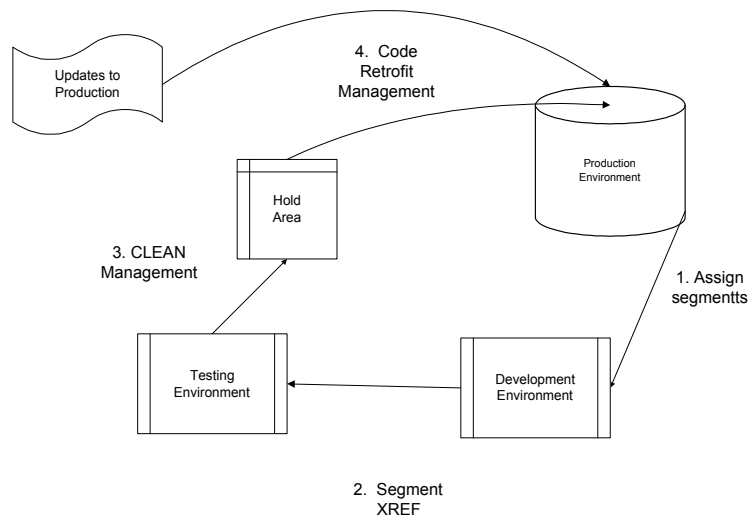


Non-Traditional model

Where the code is physically developed plays a role in determining export and import policy. In the case of code that is modified while in a change management environment, there is more control but there are also many of the same issues as the 'traditional' model. Parallel development issues still exist because typically the new code is being updated while production updates can still occur.

The Non -Traditional Code Conversion model is also comprised of several steps, each of which utilizes one or more separate product components:

Phase of Project	Related Product Components
Divide up components into segments.	Assign function
Audit Level of Change	Lines of Code Reporting Delta Sync Utility
Code Synchronization	Delta Sync Code Merging Tool (CMT)
Clean Management	Delta Sync Tagging Label Enforcement
Code Retrofit Management	Shifting Baseline

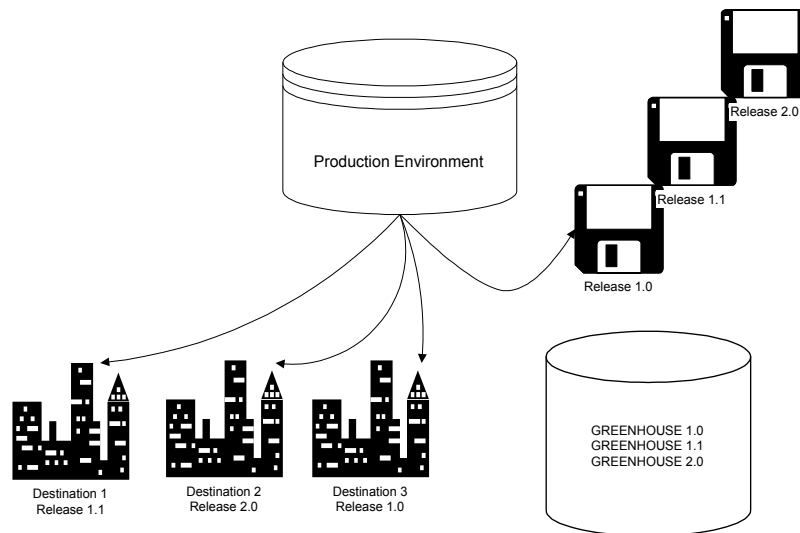


Release Management

Greenhouse enables users to build and distribute different releases of software at the same time, a situation that may occur as the result of different shipping schedules, state regulations, or a merger or acquisition.

One of the big challenges with release management is keeping track of what source has been shipped where. Many companies have core applications and then local modifications for factories, regions, etc. This presents a daunting management task for auditors and maintenance. With Greenhouse, users can label and track releases of software through the use of the functions listed below:

Phase of Project	Related Product Components
Divide up components into segments based on release or target of release (CICS region, remote hospital)	Assign function
Label releases	Tagging and standard label facility
Report on release contents	Reporting
Clean Management	Delta Sync, Tagging, and Label Enforcement
Report on collisions	Reporting and general segment management functions
Identify maintenance requirements	Reporting and general segment management functions



Summary

Greenhouse has many applications. How your company implements Greenhouse will be a function of the problem you are trying to solve. Chapter 5 of this manual outlines many of the questions you will need to ask when designing your Greenhouse implementation. There are three sets of decision and policy tables included in this chapter to provide you with guidelines for a good design session and successful implementation.

Greenhouse Concepts and Facilities

Chapter 2 : Facilities

This chapter contains:

Description of *Greenhouse Labeling Facility*.

Description of *Greenhouse Utilities*.

Description of *Lines of Code Metric Facility*.

Label Compliance Facility

Purpose : provides a mechanism for identifying portions of source inventory for special handling through the use of labels and associated Compliance Enforcement Rules.

What exactly is a label?

A label is a string of up to 60 bytes that identifies the source code in some way. You associate this string with an element (or a group of elements) using the tagging facility. Greenhouse then stores the label in its database, and the label remains associated with the element until you delete the element or you remove the label. By allowing you to assign the same label to many different elements in your inventory, this feature also allows you to group components.

What does an enforcement rule do?

Enforcement rules give you an extra measure of control over your source inventory. Instead of preventing access to entire environments, an enforcement rule allows you to be more flexible and apply protection to just the subsets of the source that you have identified as being in need of security.

How does the Label Compliance Facility work?

Using the tag utility, you assign the source components a meaningful label, which remains associated with the components. If the element being processed also matches the “For Element” clause specified in the rule, and the element in the Greenhouse database has a label that matches the label associated with the rule, then the label enforcement facility do one of the following:

Issue a warning This statement will cause the Label Enforcement processing to stop its logic checking and will allow the action to proceed with an exit return code of a four (warning).

Fail Action This statement will cause the Label Enforcement processing to stop its logic checking and will cancel the action with an exit return code of an eight (fail action).

Continue processing This statement will cause the Label Enforcement

(IGNORE) processing to stop its logic checking and will allow the action to proceed with an exit return code of zero.

For more information regarding labels and enforcement rules, refer to the *Greenhouse User Guide* or the *Greenhouse Utilities Manual*.

Greenhouse Utilities

Greenhouse Report Utility

The CIGGHUT1 utility supports several reporting functions. The Greenhouse ISPF interface can be used to generate the report syntax and JCL. Advanced functions available are additional filtering on dates or segment ID. The JCL is the same regardless of the syntax used.

Delta Sync Utility

The purpose of Delta Sync is to proactively find production elements that have changed in production as well as in the field. When an element is assigned to a segment, Greenhouse collects all valid and existing ISPF statistics. Delta Sync then compares the levels of each element to be added back with the current levels in your system. If the element has had a source change, you have two options:

1. request a report only, which will detail the change
2. request a compare and merge, which will create outputs to be passed to one of the three supported merge tools (see below)

Supported Merge tools

Greenhouse currently supports three tools for merging source code:

CMT - Merge Tool from Chicago Interface Group
PDM - Parallel Development Manager from Computer Associates
Version Merger from Data Horizons

Specific instructions regarding the CIG Merge Tool (CMT) immediately can be found in the discussion following the Delta Sync Utility in the *Greenhouse Utilities manual* as well as in the *CIG Merge Tool Reference Guide*.

Lines of Code Utility

The main purpose of the Lines of Code Utility is to provide you with a mechanism for analyzing the content of an application in terms of code, comments, and blanks. Lines of Code also provides you with the ability to analyze an application between two points in time to determine what percentage of code, comments, and blank lines have changed. This utility is typically used during testing metrics and validation of code changes.

The Tag Utility

The main purpose of the Tag Utility, CIGTAG09, is to assign labels to elements. When a label is assigned, the Greenhouse Database Log is updated with statistical information. The JCL is the same regardless of the syntax combination.

Greenhouse manages two pre-defined labels, COMPLIANT and NON-COMPLIANT. These labels are considered mutually exclusive and both cannot be current at one time. All other labels are user defined. There is no limit to the number of labels allowed per element.

Standard Label Utility

This facility, CIGGHSTD, allows you to define labels that can then be assigned to elements. By limiting the labels to a predefined list, you can implement the labeling schemes that meet your project and legal requirements.

Once standard labels are defined, all TAG requests issued from Greenhouse will have to match one of the standard labels.

Segment Control Utility

The Segment Control Utility, CIGGHGCL:

- Assigns and disables elements to a segment
- Creates a segment if it does not yet exist
- Assigns a baseline indicator for a segment

After an element is assigned, it will be considered a fully legitimate component of the segment, even though no Endeavor actions have occurred. After an element is disabled, it will not be included in segment processing, but it will still appear on the Segment Activity report.

Set baseline

Another feature of the Segment Control Utility, is the Set Baseline function, which provides the mechanism for the user to define the baseline location—key functionality for the CIGDELTA utility. This function also provides a mechanism to move the baseline location throughout a lifecycle.

Additional Utilities

Installations utilizing the Traditional method of code-conversion have access to an additional utility, the Translation Utility.

Translation Utility

When you are ready to add elements back into your source repository from external "inbound" datasets, you can either provide the SCL already prepared, or you can request that Greenhouse determine the proper inventory classification based on the data in the element list. For each element assigned to a segment, there is an entry in the element list maintained inside of the source repository. This element list contains the translation information about how to add back members from inbound datasets with the proper classification.

Lines of Code Reporting

Purpose: provides a mechanism for analyzing the content of an application in terms of code, comments, and blanks. Also provides the ability to analyze the application between two points in time to determine what percentage of code, comments, and blank lines has changed. Used during initial budget phases, code segment cost estimates, testing metrics, and validation of code changes.

The ability to analyze data from a point in time as well as between two points in time provides a strong tool for baseline metrics collection and validation of change levels. The standard Lines of Code offering provides the following key pieces of Greenhouse functionality:

- ❶ The ability to report on point in time code/blanks/comments for COBOL, Assembler, Fortran, and JCL.
- ❷ Ability to access source directly.
- ❸ Data extraction facility for building metrics data files and interfacing with other pieces of the Greenhouse product line.
- ❹ Two standard report formats:
 - ◆ Single Point in Time Analysis
 - ◆ Comparison Analysis for determining change levels

Greenhouse Concepts and Facilities

Chapter 3 : ISPF Components

This chapter contains:

Description of main applications invoked from the front end showing the main panel layouts.

The Main Menu

Purpose: The Greenhouse front end is designed to hide all complexity from the end user. Depending on the type of implementation, the user may use all or some of the functions and in various orders. Some of the functions submit batch jobs and others are online query and request functions.

The figure below shows the main panel and the functions that can be invoked.

```
----- GREENHOUSE MAIN MENU -----
OPTION ==>

Select one of the following GREENHOUSE options and press enter:

For code segment-id . . .

1 Setup          - Setup segment specific datasets for new segment
2 Build/Assign   - Build SCL or Assign syntax
3 Export         - Create, modify and submit segment export
4 Delta         - Request delta sync function
5 Reports       - Request and submit reports
6 Import        - Add back components from inbound dataset
7 Online       - View GREENHOUSE data online
8 Labeling     - Perform labeling requests
9 Clean Management - Perform clean management functions
A Complete    - Complete and cleanup the segment
D Attributes   - View and update session attributes
X EXIT        - Return to ISPF

                END = EXIT      ENTER = PROCESS      PF1 = HELP
```

Figure 1. GREENHOUSE Main Menu

Per segment, you will navigate through the application performing the following functions:

Setup

Create required control files for the segment. These files are used throughout the life of the segment until you request a COMPLETE function, which will delete the files.

Build

Register the owner of the segment, build the actual code segment, and allocate all export and WIP files for the segment.

Export

If the traditional code conversion model is being implemented, request that the segment be exported. This is facilitated through batch package SCL.

Assign

When code remediation is completed, or is not required, you can quickly and efficiently define a segment by using the Assign function. Assign enables the labeling and compliance enforcement functions of Greenhouse without stepping through the complete segment life cycle. See the Greenhouse User Guide for more information relating to the Assign function.

Delta Sync

The Delta Sync Utility is the fundamental reference point between "what was" and "what is" within a particular environment. You should use Delta Sync any time you need to determine what production changes to apply to the code exported for the segment. Delta Sync provides you with the following report and compare options:

- ❶ Request the report and compare function to show where collisions with Greenhouse segments have occurred.
- ❷ Invoke one of the supported compare and merge tools (CMT, PDM, or Version Merger).

Greenhouse Reports

There are seven standard Greenhouse reports used to list and cross-reference segment contents. These reports are submitted in batch after file tailoring from the filter panels.

Import

Once code changes are returned, request that they be added back into the system using the same segment-id. By default, Greenhouse will register the elements as 'non-compliant' until you have completed testing and reclassifying elements.

On-Line Viewing

This function is used to view Greenhouse data online.

Labeling

Once changes have been tested, the owner of a segment can tag it with the 'compliant' label or with another meaningful label. This tagging can be on an element-by-element basis, all, or part of a segment. If users have defined standard labels to the system, then Greenhouse will force the selection of a standard label.

Clean Management

This function provides both online and batch support for label activity analysis, standard label maintenance, and enforcement violations reporting. It provides both segment level and global access to the labeling data.

Complete

Once the changes have been migrated back into production, the complete function deletes all work and export files associated with the segment. It will also delete the related package from Endeavor. Greenhouse, however, will still contain all audit log data associated with the segment, and you will still be able to view and report on the completed segment.

Attributes

This function is for setting or overriding various session attributes, such as merge tool selection, high level and second level qualifiers, and work data set names and members. These options will typically be set up by the installer during systems integration. This panel allows users to override the defaults for your system.

Standard Build or Assign Function

The figure below is the standard build panel, which allows you to request the build or assign functions, choose to register owner information, and also choose to have Greenhouse allocate the export files. (Note that the Assign function is documented separately). Regardless of the owner and export dataset options, Greenhouse will eventually prompt you to construct a code segment or a list of programs. The purpose of a code segment is to provide a mechanism to collect and save lists of program names and copybook names that make up a unit of work. The code segment can then be accessed repeatedly for conversion purposes.

```
----- Build New Segment For Conversion -----
Option ==>

For segment-id . . .

Record segment owner . . . N (Y/N)
Create export files. . . . Y (Y/N)

Target dataset for SCL creation:

SCL dataset . . . CIGT.TEST.PDS
SCL member. . . . DEMO

End = Exit Enter = Process PF1 = Help
```

Figure 2. Build New Segment Panel

This panel is invoked if you respond 'Y' to Record Segment Owner on the Build panel. The option requires the setup PDS to be allocated.

```
----- VENDOR INFORMATION FOR CONVERSION -----
COMMAND ==> _____

ENTER VENDOR INFORMATION (WHERE SOURCE CODE WILL BE SENT)
FOR SEGMENT-ID: CONNIE

NAME: SILVER BULLET SOFTWARE
ADDR1: 368 W. HURON
ADDR2: SUITE 2N
CITY: CHICAGO STATE: _IL_ ZIP: ___60610_____
COUNTRY: USA
CONTACT NAME: TIM CURRIE _____
MAIL STOP: _____
PHONE1: _1-312-337-3709_____
PHONE2: _____
EMAIL ADDR: _____
```

Figure 3. Vendor Information Panel

Populating a Code Segment

Once you return from the vendor panel, Greenhouse invokes the SCL build process. Below is the Greenhouse Code Segment Create and Update panel. From here you will create a list of elements to be contained in the code segment.

```
----- Code Segment Create and Update ----- Row 1 to 11 of 119
Option ==>                                     Scroll ==> CSR
                                           Enter "/" to select option:
List Member . . BETA*           List Type. E (E/X/B) / Filters _ Append
Where Component.              Comp Type. I (I/O/P/X)
=====
Enter "/" to select function below or line command:
_ Group action | _ Msgs _ JCL _ Reports _ Preferences
=====
"/" Element           Type      Environ  S System  Subsys  (VV.LL)
-----
```

Figure 4. Code Segment Create and Update Panel

Standard Export Function

If you are exporting code via package processing, once you have completed the Setup and Build Functions, you are ready to export the code. The panel below shows the standard create function. The result of this request will be a batch package function submitted from Greenhouse.

Note that you can choose to execute the segment at this time or to only go as far as cast.

```
----- Create or Modify Segment -----
COMMAND ==>

Enter the segment name and description and the name of the dataset
containing the prebuilt code segment SCL.  If the segment already
exists, then the segment will be reset, modified, cast, and executed.

If this is the initial segment export, a new segment will be created.

For segment-id. . . . DEMOSEG1

Perform actual execution at this time? . . . _ (Y/N)

Description. . . . _____

SCL Dataset. . . . CIGT.TEST.PDS
SCL Member . . . . DAVE

End = Exit   Enter = Process   PF1 = Help
```

Figure 5. Create or Modify Segment panel

Assign Function

Assign Function

When to Perform the Assign function

The Assign function is used to group and assign elements and/or members to segments in Greenhouse, and can be used in place of package processing, and in place of Build and Export. After an element is assigned, it will be considered a fully legitimate component of the segment.

Why use the Assign function?

Clean Management:

Assign enables the labeling and compliance enforcement functions of Greenhouse without stepping through the complete segment life cycle. Once Greenhouse displays a list of elements, Assign can be invoked as a group action.

```
----- Code Segment Create and Update -----
O |----- Group Mode Action Request -----|----- Scroll ==> PAGE
| Option ==> | to select option: |
| | rs | Append |
|-----|-----|-----|
| Actions: | Execution Options: |
| 10 1. Add.....A | 1 1. Batch |
| 2. Update.....U | 2. Batch packages |
| 3. Retrieve....R | 3. Foreground |
| 4. Generate....G | 4. SCL only |
| 5. Move.....O | |
| 6. Delete.....# | Enter "/" to select option: |
| 7. Transfer....T | Edit SCL |
| 8. Signin.....SI | / Option Prompt |
| 9. Search.....SR | - Edit JCL |
|-----|-----|-----|
| GREENHOUSE Actions: | |
| 10. Assign .....AS (Batch only) | |
| 11. Disable.....DS (Batch only) | |
```

Figure 6. Invoking the Assign function

Delta Sync Function

The Delta Sync Utility is the fundamental reference point between "what was" and "what is" within a particular environment. Delta Sync is typically used to examine segment contents in order to determine what production changes need to be applied to the code exported for the segment.

From the Delta Sync panel, users have the following compare options:

- ❶ A report and compare function to show where collisions with Greenhouse segments have occurred.
- ❷ Invocation of one of the supported compare and merge tools (CMT, PDM, or Version Merger)

```
----- Endeavor Based Delta Sync Criteria -----
Command ===>

For Segment-id. . . . DEMOSEG1      Baseline name . . . _____

Compare the GREENHOUSE baseline:  Enter '/' to select options:
  To Env. . . . PROD                _ Compare VV.LL only
  Stage # . . . 2                   _ Compare source date only

If member retrofits are found:
Invoke Merge . /   Enter '/' to build the CMT merge syntax and JCL

Fill in to create Endeavor SCL. (Optional, requires CIGBST01 DD)
Action . . . . _____
CCID . . . . _____
Comment. . . . _____
Option1. . . . _____
Option2. . . . _____
Option3. . . . _____
```

Figure 7. Endeavor Delta Sync Criteria panel

Standard Report Function

Throughout the entire life of the segment, you can request one of seven available reports. The figure below shows the report selection menu. All reports accept wild carding and all will submit a batch job.

```
----- GREENHOUSE Report Menu -----
Option ==>

For Segment-id. . . . DEMOSEG1

Select one of the following GREENHOUSE reports and press enter

1 Activity      - Request a segment activity report
2 Status       - Request a segment status report
3 Content       - Request a segment contents report
4 Invoice       - Request a segment invoice report
5 Compliant     - Request a segment compliant report
6 Loc metrics   - Request a lines of code metrics report
7 Label audit   - Request a segment label audit report

End = Exit    Enter = Process    PF1 = Help
```

Figure 8.Greenhouse Report menu

Standard Import Function

Once code changes are returned, it is recommended that they be added back into the system using the same segment-id. You can add back an entire segment or only certain types from certain datasets. By default, Greenhouse will register the imported back elements as 'non-compliant' until testing has been completed and you have re-classified the elements.

```
----- Add Elements Back From Inbound DSNS -----
Option ==>

For segment-id . . . DEMOSEG1

Include ELIST overrides . . . _ (Y/N)

Optional overrides to ELIST:
Inbound DSN. . . . _____
Maps to ENV. . . . _____
      SYS. . . . _____
      SUB. . . . _____
      TYPE . . . _____

Options info for generated SCL:
Target ENV . . . . _____
CCID . . . . . _____
COMMENT. . . . . _____
```

Figure 9. Standard Addback panel

Standard Viewing Function

Throughout the entire life of the segment, you can request to view the contents, status, or owner data online. The figure below shows the online viewing panel. If you provide a segment value prior to selecting the online function, Greenhouse will display a list on the table panel, from which you can invoke one of five line commands for the segment id.

```
----- View Segment Status and Contents -----
Command ==>                               Scroll ==> CSR
List segment-id. DEMOSEG1_____ / Enter '/' for filter prompt

Enter '/' in the line command for a list of options.
-----
'/' Segment id      Exec status   Description
-----
REL0222             DEFINED      REL 2.2.2 OF ACCOUNT RECON SOFTWARE
ASSIGN6             IN-EDIT     GREENHOUSE SEGMENT DEFAULT COMMENT
ENFORCE             DEFINED      NEW SECURITY RULES
LL FRIDAY1           DEFINED      FRIDAY NIGHT EMERGENCY TURNOVER
FRIDAY2             IN-EDIT     FRIDAY NIGHT BATCH TURNOVER
FRIDAY3             DEFINED      FRIDAY NIGHT ONLINE TURNOVER
```

Figure 10. Online Viewing Menu

Once Greenhouse displays a list, you can invoke one of five line commands for the segment id, as shown above:

```
LL - Activity Log      LA - Last Actions
LC - List Contents    ST - Status
OD - Owner Data
```

Labeling and Tagging Elements

At any point in a project or segment lifecycle, elements can be labeled as compliant or some other meaningful label. The tagging function provides the mechanism to tag all or part of a segment with a compliant or non-compliant label, a free form label, or a Standard User Label. When an element is tagged on behalf of a segment, the audit log will contain an audit record stating the user, date, element, and tag requested. The Label Segment and Contents panel is shown below.

```
----- Label Segment and Contents -----
COMMAND ==>

Use this panel to set segment components compliant, non-compliant,
or equal to a user supplied value.  If standard labels are in effect
the only valid choice will be from the standard labels list.  In this case
you will be prompted with the list of standard labels to choose from.

All segment components will be labeled as per request, unless inventory
filters are provided.

For segment-id.. . DEMOSEG1

Label value..... . _____

Enter '/' to select options below:
-   Compliant                Use Compliant label
-   Non-compliant           Use Non-compliant label
-   Segment level          Create a segment level label
                               (ignores all inventory settings)
/   Filters                 Prompt for inventory filters
```

Figure 11. Label Segment Contents

If standard labels exist as part of your Greenhouse implementation, you will be prompted with the Standard Label Selection List panel, shown below.

```
----- Standard Label Selection List ----- Row 1 to 5 of 5
OPTION ==>                                SCROLL ==> CSR
Below is the list of labels currently defined as standard labels for your
implementation. These are the only valid labels that will be accepted
by the tag utility.

To select a standard label, enter 's' in the line command area and
press enter.  You will be returned to the main labeling panel.

To return without selecting a label, press PF3.

-----
Standard Labels Defined to GREENHOUSE
-----
BASELINED FOR CONVERSION
MISSION CRITICAL DATE MODULES
SHIPPED FOR RENOVATION
WARP TESTED ON LPAR 001
YEAR 2000 READY
***** Bottom of data *****
```

Figure 12. Standard Label Selection List

Selection List Function

Whenever you enter a function that does not allow wild carded values as input, Greenhouse will display the segment selection panel, shown below. From this panel you can perform all normal viewing functions as well as select a segment for batch function processing.

```
----- Select a Segment-id or View Segment Data - Row 1 to 5 of 5
Command ==>                                     Scroll ==> PAGE
List segment . . * _____ Filter . . / '/' For filter prompt

Enter '/' in the line command for a list of options or 's' to select
segment-id value.
-----
'/' Segment-id      Exec status   Description
-----
G9705A             DEFINED      GREENHOUSE SEGMENT DEFAULT COMMENT
G9705B             DEFINED      GREENHOUSE SEGMENT DEFAULT COMMENT
G9705C             DEFINED      GREENHOUSE SEGMENT DEFAULT COMMENT
G9705D             DEFINED      GREENHOUSE SEGMENT DEFAULT COMMENT
G9705E             DEFINED      GREENHOUSE SEGMENT DEFAULT COMMENT
***** Bottom of data*****
```

Figure 13. Segment Selection Panel

JCL Edit and Submit Function

Whenever you enter a function that submits a batch job, you will first be prompted with the Edit and Submit Panel, shown below. From this panel you can edit the JCL created during file tailoring, submit the JCL, and change job card settings.

```
----- GREENHOUSE Edit and Submit Menu -----
OPTION ==>

Select one of the following options and press enter:

                1  Edit JCL
                2  Submit JCL

Dataset and member for SCL build:
SCL Dsn . . . CIGT.TEST.PDS
Member. . . . DAVE

Job cards for all file tailoring:
. . . . //HV007TAG JOB  (04330),'CIG-INC',CLASS=2,REGION=4096K,
. . . . //              MSGCLASS=H,MSGLEVEL=(1,1)
. . . . _____
```

Figure 14. GREENHOUSE Edit and Submit Menu

There are several more pop-up filtering panels, warning boxes, and tutorials. For more information on any ISPF functions, refer to the *Greenhouse User Guide*. For more information on each of the utilities discussed in this chapter, refer to the *Greenhouse Utilities manual*.

Greenhouse Concepts and Facilities

Chapter 4 : Reports

This chapter introduces all Greenhouse specific reports.

GREENHOUSE Reports

As mentioned in the previous chapter, Greenhouse provides a choice of seven packaged reports, which are described in the following pages:

- ❶ Activity
- ❷ Status
- ❸ Content
- ❹ Invoice
- ❺ Compliance
- ❻ Lines of Code Metrics
- ❼ Label Audit

Invoking Reports

The standard way to invoke the reports is through a batch job created by the ISPF front-end. The ISPF application builds all syntax and JCL streams through standard ISPF file tailoring processes. Option 5 from the main menu provides a sub-menu for report selection. See the *Greenhouse User Guide* for more information about invoking each report from the ISPF front-end.

Activity Report

Purpose: Provides you with one audit trail from start to finish of the code segment. Records all iterations of element activity, package activity, and warning messages issued to the users of the segment.

Usage: Whenever there is any segment activity or by request

Inputs: Package Audit Log Data and User Syntax

Segment Status Report

Purpose: Provides management with a quantified status of each segment. From this report you can determine number of elements in a segment, how long the segment has been out, and what number and percentage have been added back.

Usage: Weekly for management meetings or by request

Inputs: Package File and User Syntax

Code Segment Contents Report

Purpose: Provides segment managers a list of all elements that were acted upon in the last execution of package. It shows collisions and status of executions, and provides a cross-reference of this report based on element key versus segment. This report not only lists the elements in the package, but also points out where there are collisions with other packages. This collision management is a standard feature of the product and requires no setup.

Usage: After each segment execution and weekly for cross-reference

Inputs: Package File and User Syntax

Segment Invoice Report

Purpose: Documents the contents of a segment in terms of lines of code and status information. The highly detailed nature of this report makes it a perfect packing list for project management.

Inputs: User syntax request

Segment Lines of Code Report

Purpose: Provide you with a mechanism for analyzing the content of an application in terms of code, comment and blanks. Also provide the ability to analyze the application between two points in time to determine what percentage of code, comments, and blank lines have changed. Used during initial budget phases, code segment cost estimates, testing metrics, and validation of code changes.

Inputs: User syntax request

The ability to analyze data from a point in time, as well as between two points-in-time, provides a strong tool for baseline metrics collection and validation of change levels. The standard product offering provides the following key pieces of Greenhouse functionality:

- ◆ Ability to collect or report on point in time code/blanks/comment for COBOL, Assembler, PLI, Fortran, JCL, PROC, and Data types.

- ◆ Ability to access source directly.
- ◆ Data extraction facility for building metrics data files and interfacing with other pieces of the Greenhouse product line.
- ◆ Three standard report formats:
 - Single Point in Time Analysis
 - Comparison Analysis for determining change levels
 - Threshold Analysis

Label Audit Report

Whenever there is labeling on the behalf of a segment, an audit record is created. The label audit report is designed to list the activity of one or more elements/members in the area of labeling and labeling enforcement. Additionally, the report will show the sequential label history of each element and segment in the Greenhouse database.

Compliance Report

A Compliance Report returns a list of elements in a segment that have been labeled as compliant. This report can be tailored to provide specific element location information, such as Endeavor environment, system, subsystem, type, and stage information.

Sample Reports

For sample report layouts, refer to the *Greenhouse User Guide*.

Greenhouse Concepts and Facilities

Chapter 5 : Getting Started

This chapter includes Implementation Objectives and Decision Policy and Design Tables for:

- Clean Management
- Release Management
- Traditional and Non-Traditional Code Conversion Models

Implementation Objectives for Clean Management

The five major objectives in this kind of Greenhouse implementation are the same regardless of the source management tool employed.

❶	Maintain knowledge of activity against select components throughout the life cycle.	For each component, you can assign one or more labels that will then be used as a trigger during various exit and automation points. The knowledge of component activity is logged in the Greenhouse data base and is then available in reports or on line queries.
❷	Restrict access or update	Based on the assign label, Greenhouse will ignore, warn or prohibit various actions against components identified in the enforce rule set. This includes retrieving back levels of code that are not 'certified', for instance.
❸	Leverage current configuration and inventory assignment knowledge.	You have already done a lot of work to classify your source components and relationships. Greenhouse allows to you assign components to segments and then reference them using the various source management nomenclatures. Endeavor components are an example of these tools.
❹	Provide various levels of metrics and status reporting to management.	Management needs concise summary level data that reports status and numbers. Greenhouse provides senior, mid-level and line managers with the type of data they need to support your efforts and effectively gauge the Clean Management effort.
❺	Audit Trail	Through online and standard label audit reports, managers and auditors will be able to track due diligence down to the component level if desired.

Decision Policy and Design Tables for Clean Management

The following are some of the policy decisions your organization must face. As with the design issues, this list will cause more questions to be asked, but that is the goal.

Policy Topic	Description	Recommendation
Is there a Clean Management mandate or legal claim at your company?	Why are you implementing a Clean Management program?	
Clean Management Goals?	What do you want to accomplish with Greenhouse?	
Use of Standard Labels	What labeling scheme will accomplish the goals of the enforcement rules?	We recommend that you employ the use of the standard labeling facility to ensure that the rule usage is consistent across all users.
Inventory to monitor	Do you want to monitor all activity or activity only on those components that are key to an implementation?	
Life cycle issues	Where do you want to begin the enforcement process?	
How strict will enforcement be?	Will you log, warn or prohibit actions?	You can have a combination of consequences based on inventory location and label values.
Which projects will use labeling?	Will Greenhouse be implemented only for a special project, Y2K for example, or will it be used for the all projects?	
Will labels be used for release management as well as enforcement?	This is a consideration because your label/inventory combination may limit access to code.	
Who will maintain the enforcement rules?	This is a policy and security question.	
Who will maintain the assign and disable functions of standard labels?	This is a policy and security question.	
Will testing labels be included in the process?	Since label usage is not limited to production or certified, many customers are labeling components based on the level of testing that has been successfully performed.	

Design Decisions for Clean Management

The following table is a list of typical design decisions that will have to be made. In most cases, these questions will elicit more questions. Expect this to be the case until the design is finalized. The goal of the list is to get you and your team thinking and talking about these topics.

Design Topic	Description	Recommendation
Segment Naming Standards	For those customers using package processing, the segment-id can be used as the package-id.	None.
Enforcement	What enforcement will be put into place to guarantee usage of system?	
Who will assign the components to a segment?	Will the administrator or end user assign components to code segments?	
Who will use the online portion of the product?	This affects the level of training required.	
Will the product be used for more than one purpose?	Again this affects who is using the product and who will have to be trained.	
How will segments be organized?	This aspect is very fluid. Endeavor implementations may want to organize using package processing or inventory groups.	

Implementation Objectives for Release Management

The five major objectives in this kind of Greenhouse implementation are the same regardless of source management tool employed.

❶	Maintain the knowledge of what was contained in every release to every location.	For each software release to each target, segment data will track the contents of release.
❷	Understand where software releases have collisions.	For each segment, a cross-reference is available that will show where other segments (releases) have the same component but at a different level or the same component.
❸	Leverage current configuration and inventory assignment knowledge.	You have already done a lot of work to classify your source components and relationships. Greenhouse allows to you assign components to segments and then reference them using the various source management nomenclatures. Endeavor components are an example of these tools.
❹	Provide various levels of metrics and status reporting to management.	Management needs concise summary level data that reports status and numbers. Greenhouse provides senior, mid-level and line managers with the type of data they need to support your efforts and effectively gauge the Clean Management effort.
❺	Audit Trail	Through online and standard label audit reports, managers and auditors will be able to track due diligence down to the component level if desired.

Decision Policy and Design Tables for Release Management

The following are some of the policy decisions your organization must face. As with the design issues, this list will cause more questions to be asked, but that is the goal.

Policy Topic	Description	Recommendation
Who needs the Release Management information?	Managers will need different data than the maintenance programmers.	
What problems are you trying to solve?	Knowing the problem ensures that we build the correct solution set.	
Use of Standard Labels	Will labeling also be used for reporting and release naming purposes.	
Releases to monitor	Do you want to track all software releases or only major ones?	
Target locations to monitor.	Do you want to track all target locations or only the problematic ones?	
How will the assignment of segments take place.	As a batch job? In the foreground.	
When is a release defined?	Do you know the contents of a release at the start of a development cycle or only at the end?	
Who will maintain the assignment of segments?	This is a policy and security question.	
Do you ship maintenance as a release or as individual fixes?	Greenhouse will know what levels of software are contained in each release. This is valuable information for the maintenance programmers. Is there also a regular maintenance release shipped?	
Would individual fixes be tracked by GREENHOUSE?		

Design Decisions for Release Management

The following table is a list of typical design decisions that will have to be made. In most cases, these questions will elicit more questions. Expect this to be the case until the design is finalized. The goal of the list is to get you and your team thinking and talking about these topics.

Design Topic	Description	Recommendation
Segment Naming Standards	This all depends on the problem you are solving. If the problem is that there are several targets each with different levels of software, maybe your segment ids should be the target destinations. If the problem is tracking changes per software release, maybe your segment ids should be tape numbers or release numbers.	None.
Enforcement	What enforcement will be put into place to guarantee usage of system?	
Who will assign the components to a segment?	Will the administrator or end user assign components to code segments?	
Who will use the online portion of the product?	This affects the level of training required.	
Will the product be used for more than one purpose?	Again this affects who is using the product and who will have to be trained.	
How will segments be organized?	This aspect is very fluid. Endeavor implementations may want to organize using package processing or inventory groups.	

Implementation Objectives for Traditional and Non Traditional Code Conversion Models

The five major objectives in this kind of Greenhouse implementation exist whether the code changes are being made inside or outside of your organization.

❶	Maintain knowledge of segment contents and status throughout the life of the segment.	For each segment of work that gets exported outside of production, you want to know what elements, at what level, are in the segment. Additionally, you want lines of code metrics and knowledge of when elements are being added back into the system.
❷	Maintain knowledge of segment contents and status throughout the life of the segment.	For each segment of work, you need to know the elements that will need to be reviewed for code retrofits. This is a standard parallel development problem. As code is taken out of production for short or long term projects, there is always the chance that the code in production will be modified through emergency or previously scheduled paths. The net result is that the baseline for the code exported from the system is no longer in sync, and someone will have to retrofit production changes or at least make a judgment call as to whether the code needs to be retrofitted.
❸	Leverage current configuration and inventory assignment knowledge.	You have already done a lot of work to classify your source components and relationships. Greenhouse allows to you build code segments based on stored application knowledge and maintain a sealed method of mapping elements back into Endeavor, which ensures type, subsystem, and system integrity
❹	Provide various levels of metrics and status reporting to management.	Management needs concise summary level data that reports status and numbers. Greenhouse provides senior, mid-level and line managers with the type of data they need to support your efforts and effectively gauge the status of each code segment and thus the status of the overall effort.
❺	Labeling and audit logs.	Integrated with Greenhouse is the ability to identify portions of source inventory for special handling through the use of labels and associated Compliance Enforcement Rules. Through the use of such labels, Greenhouse provides proof that the project took place, was audited, and also keeps track of who made decisions along the way.

Design Decisions for Traditional Code Conversion Implementations

The following table is a list of typical design decisions that will have to be made. In most cases, these questions will elicit more questions. Expect this to be the case until the design is finalized. The goal of the list is to get you and your team thinking and talking about these topics.

Design Topic	Description	Recommendation
Dataset Naming Standards	Code segments will be exported out of your source repository into various PDSs. There needs to be a naming standard for the outbound as well as the inbound datasets.	Utilize the source location names, plus the segment name as nodes in the PDS. This will make dataset population easy through a processor. Also use the last node to signify 'out' or 'in'.
Segment Naming Standards	The segment name is used as the package id internally, if package processing is employed.	Limit to eight characters, using first two to signify they are for the Year 2000 conversion.
Security for outbound datasets	Will the outbound datasets be "read only" except for Endeavor?	Recommend "read only" to maintain baseline image.
Export or outbound PDS population	How will the source be exported outside of Endeavor control?	If the code is being exported out of Endeavor, then either the Retrieve or Generate action would suffice.
Enforcement	What enforcement will be put into place to guarantee usage of system?	Minimally force the usage of packages for all environment activities, for both stages. The package reuse feature will allow the same id to be used throughout the entire life of the package/segment.

Table of Figures

Figure 1.	GREENHOUSE Main Menu	3-3
Figure 2.	Build New Segment Panel	3-6
Figure 3.	Vendor Information Panel	3-6
Figure 4.	Code Segment Create and Update Panel	3-7
Figure 5.	Create or Modify Segment panel	3-8
Figure 6.	Invoking the Assign function	3-9
Figure 7.	Endevor Delta Sync Criteria panel	3-10
Figure 8.	Greenhouse Report menu	3-11
Figure 9.	Standard Addback panel	3-12
Figure 10.	Online Viewing Menu	3-13
Figure 11.	Label Segment Contents	3-14
Figure 12.	Standard Label Selection List	3-14
Figure 13.	Segment Selection Panel	3-15
Figure 14.	GREENHOUSE Edit and Submit Menu	3-16