CA Endevor® Software Change Manager

Best Practices Guide

Version 17.0.00

CA Technologies Product References

This document references the following CA Technologies products:

- CA Common Services™ (CA Common Services)
- CA Endevor® Software Change Manager (CA Endevor SCM)
- CA InterTest™ Batch (CA InterTest Batch)
- CA Librarian® Base for z/OS (CA Librarian Base for z/OS)
- CA Chorus™ Software Manager (CA CSM)
- CA Panvalet® for z/OS (CA Panvalet for z/OS)
- CA Service Desk Manager (CA Service Desk Manager)

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Best Practices Guide Process

These best practices are based on customer experience reported through interviews with development, technical support, and technical services. Therefore, many of these best practices are a collaborative effort stemming from customer feedback.

To continue to build on this process, we encourage you to share common themes of product use that might benefit other users. Please consider sharing your best practices with us.

To share your best *practices, c*ontact us at techpubs@ca.com and preface your email subject line with "Best Practices for product name" so that we can easily identify and categorize them.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

Version 16, Second Edition

- Introduction (see page 9)— Removed redundant information.

Version 16

- First Time Installation Using the BPI (see page 13)— Added to recommend the Best Practice Implementation method for initial installations.
- Review Optional Features Table (see page 29)— Updated to recommend the following options: DEFAULT_REVERSE, ELMNM_PROMPT, INTRDR_ALTID, PKGID_PROMPT, and RESERVED_OTHERDSN.

Contents

Chapter 1: Introduction

The guide introduces the CA Technologies mainframe management strategy and features, and describes the best practices for installing and configuring your product.

The intended audience of this guide is systems programmers and administrators who install, maintain, deploy, and configure your product.

Chapter 2: Configuration Best Practices

This chapter describes best practices to configure CA Endevor SCM for optimal performance in meeting the specific needs of your site's software change management activities and for ease of use.

This section contains the following topics:

Implement a Proactive Preventive Maintenance Strategy

CA Technologies formerly delivered product maintenance using Service Packs. We have replaced this model with CA Recommended Service (CA RS) for z/OS, which provides more flexibility and granular application intervals. CA RS is patterned after the IBM preventive maintenance model, Recommended Service Upgrade (RSU). With CA RS, you can install preventive maintenance for most CA Technologies z/OS-based products in a consistent way on a schedule that you select (for example, monthly, quarterly, annually).

We recommend that you develop and implement a proactive preventive maintenance strategy whereby you regularly apply maintenance. You could follow the same schedule that you use to apply IBM maintenance, or you could implement a schedule for CA Technologies products only.

Business Value:

Keeping your products current with maintenance helps your team remain productive and minimize errors while safely protecting your systems. If you do not install preventive maintenance regularly, you risk encountering known problems for which we have published and tested fixes.

Our mainframe maintenance philosophy is predicated upon granting you the flexibility to maintain your sites and systems consistent with industry best practices and site-specific requirements. Our philosophy focuses on two maintenance types. Understanding each type can help you maintain your systems in the most efficient manner.

Note: This philosophy applies to the CA Chorus Software Manager Enabled Products. For legacy products, contact CA Support for maintenance details.

Corrective Maintenance

Helps you address a specific and immediate issue. This type of maintenance is necessary after you encounter a problem. We may provide a test APAR when a new problem is uncovered, or a confirmed PTF when the problem has been resolved. Your primary goal is to return your system to the same functional state that it was before you experienced the issue. This type of maintenance is applied on an as-needed basis.

Preventive Maintenance

Lets you apply PTFs that we have created and made public. You may have experienced the issues that each PTF addresses. CA RS provides a way to identify all published maintenance that has been successfully integration-tested. This maintenance has been tested with other CA Technologies products, current z/OS releases, and IBM subsystems, such as CICS and DB2. CA RS levels are published monthly that include PTFs, HIPERs and PRPs (PE-resolving PTFs). Before you download, apply, and test a new CA RS level, we recommend that you accept the previous CA RS level.

You can initiate a maintenance installation activity at any time. You can then install the current CA RS level of maintenance (recommended) or an earlier level. Additionally, you can install maintenance to support a new hardware device, software upgrade, or function using the FIXCAT method.

For all maintenance, *before* you initiate any maintenance action, obtain the current SMP/E HOLDDATA.

Important! CA Chorus™ Software Manager (CA CSM) - formerly known as CA Mainframe Software Manager™ (CA MSM) - is an intuitive web-based tool that can automate and simplify many CA Technologies product installation and maintenance activities. We strongly recommend that you use CA CSM to maintain your CA Technologies z/OS-based products.

More Information:

To apply preventive maintenance using CA CSM or from CA Support Online on http://ca.com/support, see the *Installation Guide* for your product and the CA CSM online help.

First Time Installation Using the BPI

For first-time installations, use the Best Practices Implementation (BPI) method to set up a complete implementation of the product. After installing CA Endevor SCM, use the BPI method to set up an implementation, with minimum customization, to get you started using the application. The BPI is delivered as a set of input tables and jobs that you configure and run. After you complete the BPI, you will be ready to load your source code into the CA Endevor SCM inventory structure that supports the software development lifecycle.

Business Value:

The BPI method builds a best practices and customizable software development environment that is complete and includes a life cycle and inventory structure.

More information:

For more information on the BPI process, see the "Best Practices Implementation" in the *Scenario Guide*.

Logical Structure

The following best practices apply to implementing your site's logical structure, which includes your site's software change management lifecycle and software inventory classification scheme.

Avoid Confusing Environment and Stage Names

All environment names and stage names should be the same length and the names should be unique.

Business Value:

When names are the same length, it is easier to use a name as part of a data set name. Unique names avoid confusion when referencing a particular stage.

Make System and Subsystem Names Easy to Identify

Each system name should identify the purpose of the application. Each subsystem name should identify the specific application within the system.

Business Value:

Names that are easy to identify help avoid confusion.

Additional Considerations:

If you are performing concurrent development, you can use subsystems in the development environment to manage the concurrent development.

More information:

For more information about using subsystems for concurrent development, see Use Sandboxes for Concurrent Development. (see page 16)

Normalize Type Names

Type names should be *normalized*; that is, type names should correspond to the language, but not the version, of the source code (COBOL, JCL, CLIST, and so on). For example, there should be one type for COBOL, not separate types for COBOL/LE and COBOL II. Processor groups can be used to distinguish between language versions, such as COBOL LE and COBOL 2. We recommend the following Type names:

- ASMMAC
- ASMPGM
- BIND
- CHDR
- CPGM
- CICSMAP
- COPYBOOK
- COBOL
- CLSTREXX
- DCLGEN
- DOC
- EZTINCL
- EZTRIEVE
- FORM
- FORTRAN
- ISPM
- INCLUDE
- ISPP
- ISPS
- ISPT
- JAR
- JAVA
- JCL
- LINK
- PARM
- PL1
- PROC
- PROCESS

- RULES
- SANDBOX
- SCL
- TXT
- XML

Business Value:

Normalized type names make it much easier to upgrade the code from one version of the language to another.

Use Sandboxes for Concurrent Development

Use private work areas (sandboxes) for concurrent development.

Business Value:

Sandboxes provide separate work areas for different development activities. This lets developers work on the same code simultaneously, without interfering with each other's work, thereby improving the efficiency of the development staff. Typically, several sandboxes are set up so that developers can work on the same code independently; and then, the changes made in the sandboxes are combined in an integration subsystem, where the changes can be accepted or rejected.

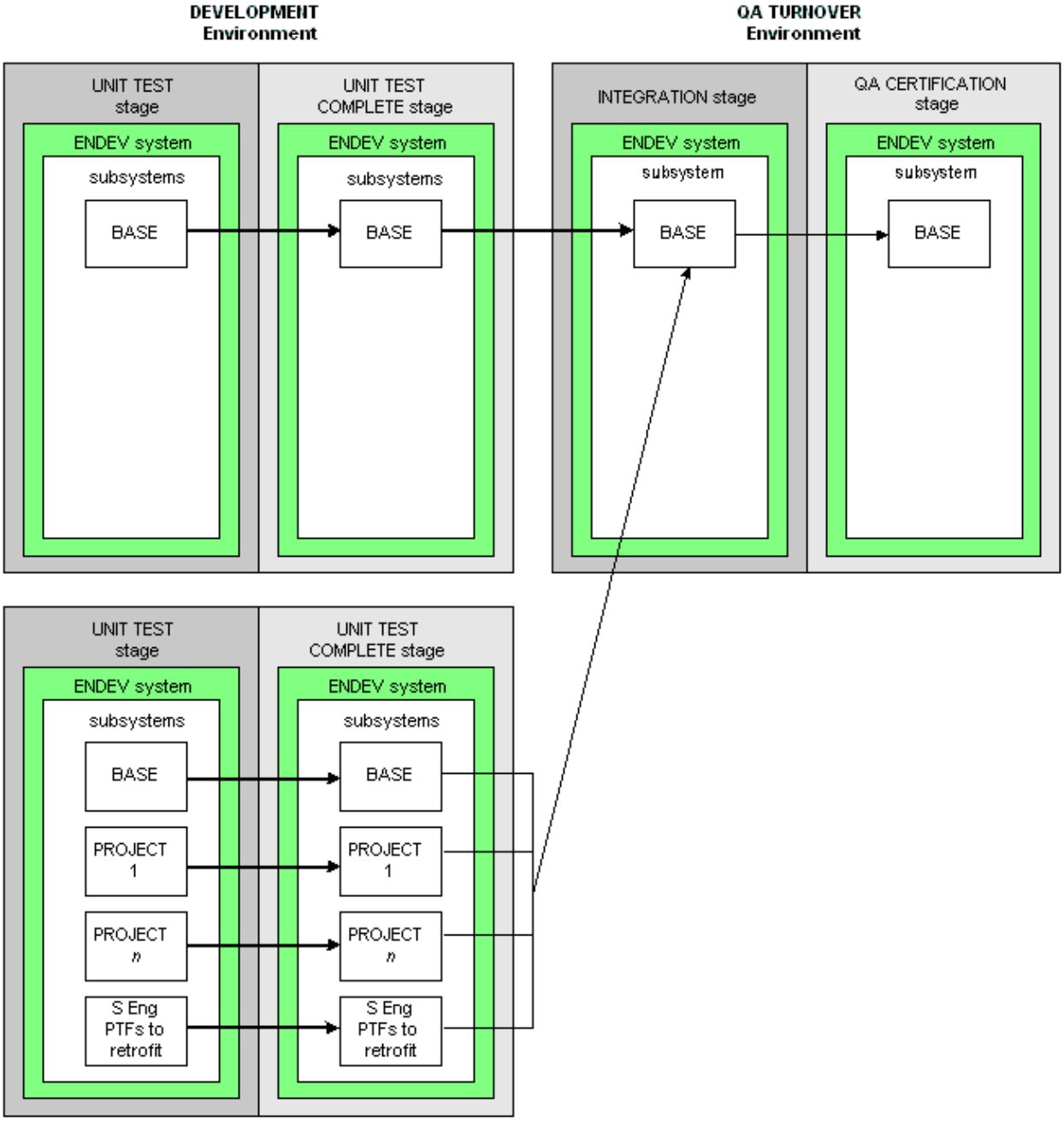
Additional Considerations:

Methods of sandbox development include: 1) subsystem sandboxes; 2) environment sandboxes; and 3) private work area stages. Subsystem sandboxes are easiest to setup; however, if subsystems are required to subdivide systems (applications), it is necessary to set up environment sandboxes using the environment sandboxes method or the private work area stages method.

Details regarding sandbox development methods follow:

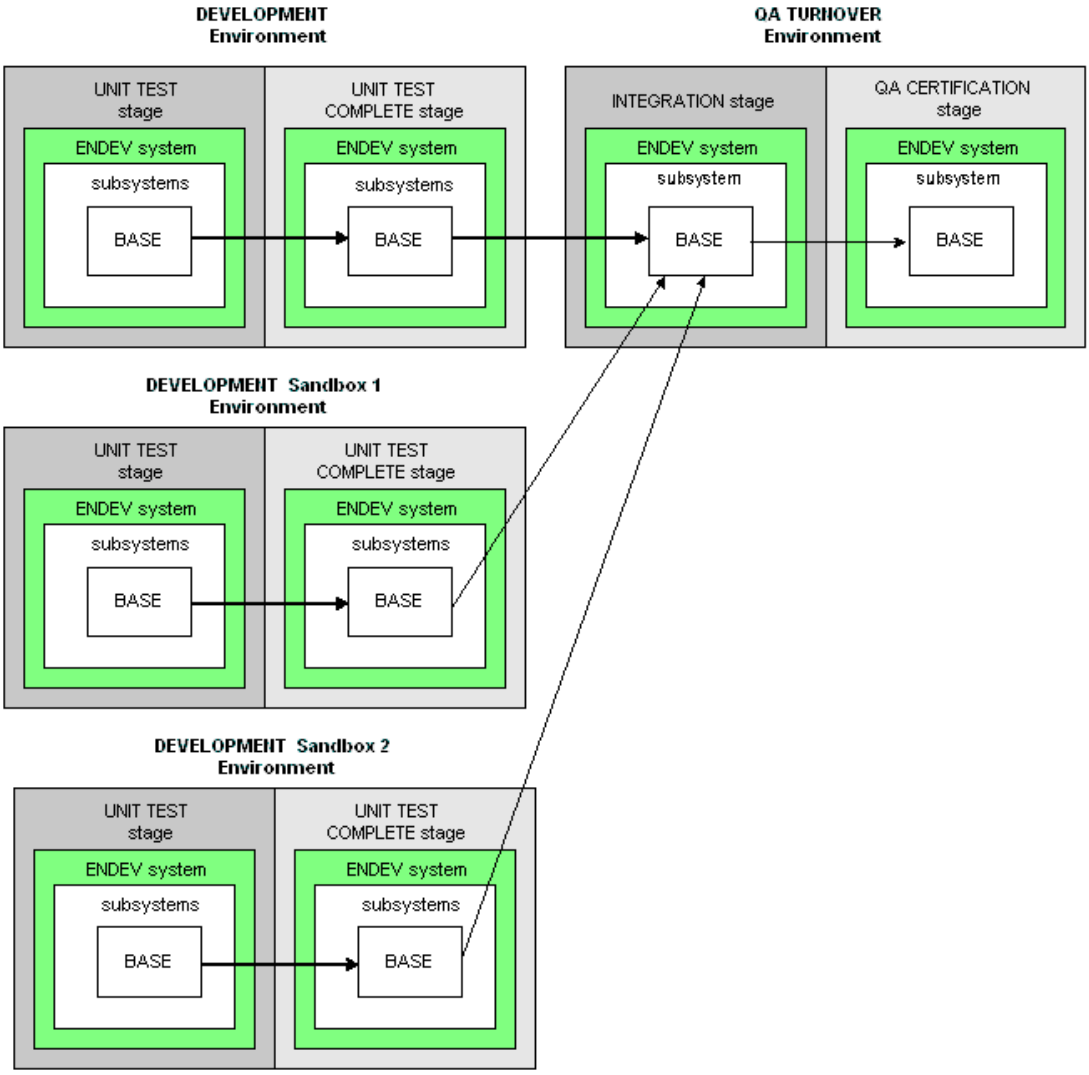
- **Subsystem sandboxes**—Use subsystem sandboxes, if you do not use subsystems for specific applications. This method is easier to set up and maintain than environment sandboxes. Adding a new environment requires the modification to and re-assembly of the C1DEFLTS table and the creation of all of the necessary environment related data sets. Adding a new subsystem consists only of adding the subsystem to the system definition, adding a few required data sets, and sharing processors. In addition, it is easier to delete a subsystem than to delete an environment.

For example, the following diagram shows development of a code branch where coding changes for projects 1 through *n* are performed in subsystem sandboxes. Also, the maintenance team has their own subsystem sandbox. Then all development changes are integrated in the Integration stage.



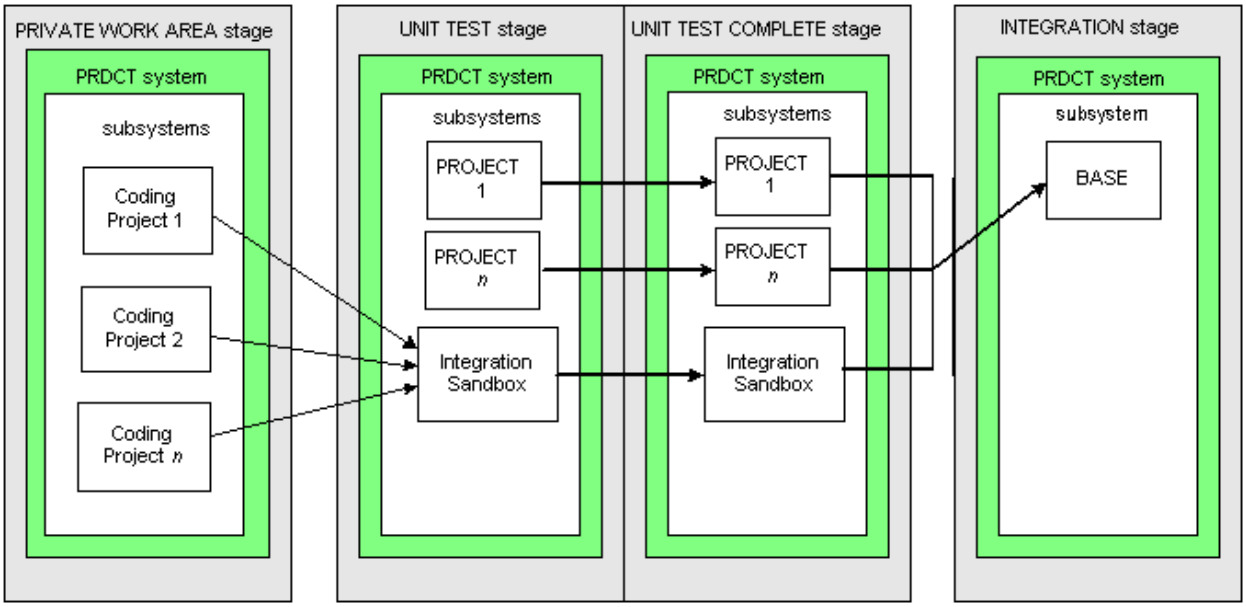
- **Environment sandboxes**—Use separate environments for your sandbox development areas, if you are using subsystems for specific applications, for example, if you have a system for your Finance department's applications that contains subsystems for your Accounts Payable and Accounts Receivables applications. The advantage of environment sandboxes is that you can use the same system and subsystem names and relationships in each sandbox environment. The disadvantage is that this method requires additional administrative effort to setup each sandbox environment, which requires the modification to and re-assembly of the C1DEFLTS table and the creation of all of the necessary environment related data sets.

  For example, this diagram shows development activities can be performed in the Development Sandbox 1 environment and the Development Sandbox 2 environment. Then the changes can be integrated in the Integration stage.



- **Private Work Area Stages**—Use an additional stage with subsystem sandboxes, if you are using subsystems for specific applications. This method requires that you set up one additional environment and stage to hold your subsystem sandboxes, and you must add an integration subsystem to a subsequent stage. The advantage is that you only need to set up one additional environment and stage combination with as many sandbox subsystems as you need, instead of the multiple environments required for the environment sandboxes method.

  For example, the following diagram shows a stage named Private Work Area that contains multiple subsystem sandboxes for development activity. The development changes are integrated in a subsystem in the Unit Test stage.



Use Subsystems to Support Multiple Releases

If you need to maintain multiple releases of the same application concurrently, use different systems (or subsystems, if available) to segregate the different versions of the code. Make sure that the system (or subsystem) name is incorporated into the related data set names to create unique libraries.

Some sites require maintaining multiple releases of the same program or element concurrently. For example, this requirement can occur when the site is a software vendor that needs to support program A shipped in release 1, release 2, and release 3, because not all of their clients have implemented the latest release.

Business Value:

The subsystem definition provides the flexibility most sites are looking for in this situation. Using the application name as the system definition and the release number as the subsystem definition has the following advantages:

- The CA Endevor SCM administrator needs to only maintain one set of definitions within CA Endevor SCM because the majority of administration is at the system level.
- You are exploiting the ability of CA Endevor SCM to logically categorize elements under its control.
- You can extend the definition into the physical definition to create mutually exclusive physical inventories.
- You can introduce new releases merely by adding new subsystems.

Physical Structure

The following best practices apply to implementing the physical data set structure that supports your site's logical structure.

Recommended Data Set Format

Use PDSE data sets.

Business Value:

A PDSE is similar to a PDS, however the PDSE architecture includes dynamic expansion of the data set, and space from deleted or moved members is automatically reused for new members. This architecture minimizes the chances of an ABEND occurring because of lack of library space; therefore, it is far more efficient to use PDSE data sets than traditional PDS data sets.

Defining Base and Output Libraries

Do not share base libraries across environments, stages, systems, and types. Do not share libraries across subsystems when subsystems are used as sandbox names. These recommendations also apply to processor output libraries, such as object, load and listing libraries.

Business Value:

Having separate base and output libraries for each type avoids name conflicts, which otherwise could occur if you had two members with the same name that were different types.

Defining Delta Libraries

Define a minimum of one delta library for each stage. Use CA Endevor SCM BDAM ELIBs for the delta libraries.

Business Value:

Although delta libraries can be shared across stages in an environment, we recommend a minimum of one delta library per stage.

Binary Types

Use full-image deltas as the delta format for binary type elements.

Business Value:

Source compare is impossible or unnecessary for binary files and is not performed with the full image delta format.

Use Symbolic Variables in Data Set Names

The following best practices apply to using symbolic variables in data set names:

- Use symbolic variables to define base and delta libraries.

  For example, the following library name refers to a library called SRCLIB for each subsystem and stage combination within a given system.

  ```
  &C1SYSTEM..&C1SUBSYS..&C1STGID..SRCLIB
  ```

Business Value:

Using symbolic variables allows the same type definition to point to different libraries based on the inventory classification of the element. You can use the symbolics when specifying base and delta libraries on type definition panels. During processing, CA Endevor SCM replaces the symbolic with the proper information from the action request.

More Information:

For more information, see Using Symbolics to Define CA Endevor SCM Libraries in the *Installation Guide*.

- Use data set names that let you take advantage of the Processor Map Allocation feature. For this feature, certain symbolics can be used for the location specifications in the processor DD statement. Other symbolics are allowed, but will remain the same for all data sets in the concatenation.

  Business Value:

  The Processor Map Allocation feature provides the ability to code a varying concatenation of data sets to a single DD statement based on the map, using location (environment, stage, system, subsystem) information provided by symbolics within the data set name, eliminating the need to use if-then-else logic. This feature facilitates the implementation of subsystem sandboxes (private work areas) by eliminating the need for alias data set names when system or subsystem names change across environments. This feature works for data sets whose names follow the mapping structure of the location from which the processor is executing; therefore, processors do not have to be regenerated after a change to the mapping structure.

  Additional Considerations:

  For purposes of the Processor Map Allocation feature, the following symbolics can be used for the location specifications in the DD statement. Other symbolics are allowed, but will remain the same for all data sets in the concatenation.

Identify libraries by	Using this symbolic	Or this alias
environment	&C1ENVMNT	&C1EN
stage	&C1STAGE	&C1ST
stage ID	&C1STGID	&C1SI
stage number	&C1STGNUM	&C1S#
system	&C1SYSTEM	&C1SY
subsystem	&C1SUBSYS	&C1SU

  More Information:

  For more information, see Implement Processor Map Allocation in the *Extended Processors Guide*.

Define Site Symbolics to Reference Libraries with Nonconforming Names

Use site symbolics to reference CA Endevor SCM libraries in processors, where your libraries do not following the CA Endevor SCM naming convention, for example, CICS libraries, DB2 libraries, processor step libraries, system libraries, and so on. Site symbolics are required if you are using USS HFS pathname specifications for element type base or source output file definitions. Otherwise, site symbolics are optional.

Business Value:

The Site Symbolics facility lets you define global symbols that can be used in type and processor definitions to reference data set name specifications for base, delta, source output, include libraries, and processors. Thus, commonly referenced data sets can be defined in a single location (the Site Symbolics table) significantly easing maintenance. At execution time, all site symbolics referenced by a processor are stored with the processor symbolics in the component data. If changes are needed, the Site Symbolics table can be updated, but the processors do not need to be regenerated.

More Information:

For more information, see the chapter "Site Symbolics" in the *Administration Guide*.

Processors

You can use the processors delivered with the Best Practices Implementation for reference.

The following best practices apply to processor JCL:

- Use symbolics in your processors to reference data sets.

  Business Value:

  With symbolics, one physical processor can support multiple processor groups. Symbolic overrides can be very valuable in providing flexibility to processors. They can support variations in compile and link parameters and in the inclusion or exclusion of steps such as adding or bypassing testing tools such as CA InterTest Batch.

  Additional Considerations:

  Avoid having an excessive number of processor groups. When the number of processor groups becomes unwieldy an alternative approach to supporting some of the functions (such as compile and link parameter substitutions) should be used.

- Use the LIBENV option on the C1DEFLTS table to enable -INC or ++INCLUDE syntax in processors for code that is used in multiple processor locations. The LIBENV parameter does not indicate that CA Librarian or CA Panvalet is used; it merely indicates which INCLUDE syntax style is used.

CA Endevor SCM determines which syntax to resolve, based on the definition of the field LIBENV in the C1DEFTLS table. This value can be set to either PV (for CA Panvalet syntax) or LB (for CA Librarian syntax). PV resolves ++INCLUDE statements and LB resolves –INC statements. You can set these values even if you do not have the CA Panvalet or CA Librarian CA Endevor SCM interfaces, because the purpose of LIBENV is only to identify to CA Endevor SCM how to resolve the syntax when it is encountered in element source.

Business Value:

Using INCLUDE statements makes it easier and quicker to create new processors as new languages and processing requirements arise at your site. You can exploit INCLUDE statements in CA Endevor SCM processors to clean up the look of the processor and to share commonly used routines across various processors. INCLUDE statements read from a library and insert commonly used JCL into processors.

Additional Considerations:

You can also imbed INCLUDE statements within other INCLUDE statements to expand the modularity of your routines.

- Use consistent processor symbolic variable names across processors.

  Business Value:

  Using consistent processor symbolic variable names make the maintenance of processors easier, because there is no need to constantly ask yourself what the variable is referring to. The consistent naming of variables in processors is as valuable to the maintenance function as is the consistent naming of variables in programs.

  Additional Considerations:

  You can use an INCLUDE statement in your processor to insert a member that contains the symbolic variable names.

- Use site symbolics as a method of reducing or eliminating processor group overrides.

  Business Value:

  The Site Symbolics table provides the ability to override symbolics without having to modify a large number of processor groups.

- Use CA Endevor SCM utilities before using IBM utilities in executing processor steps.

  Business Value:

  The CA Endevor SCM utilities are specifically designed to work with CA Endevor SCM within processors. These utilities support code (such as footprints) that may not be supported by other utilities and they capitalize on internal CA Endevor SCM functionality that is foreign to other utilities.

Additional Considerations:

IBM utilities can be used where CA Endevor SCM utilities are not available to perform the required function. IBM utilities are generally more reliable and more consistently updated within the operating system than are those of third parties.

- Other best practices for processors:
  - Balance the number of processors and the complexity level of the processors with the technical skills of the individuals who must support them. With an experienced staff, you can use one very complex COBOL generate processor that supports batch, CICS, DB2, and CA InterTest Batch. With a less experienced staff, you would need multiple processors that are less complicated, such as separate processors that support batch, CICS, DB2, and CA InterTest Batch. Multiple Processors support more limited functionality but are easier to understand and maintain.
  - Label or indent IF/ELSE/ENDIF statements to make them more readable and their ranges more identifiable.
  - Comment the steps in your processors to make it easy to understand the purpose of each step.
  - Each language level should have its own generate processor to reduce the complexity of each processor and to make it easier to retire obsolete levels. In addition, one move processor and one delete processor are recommended for each type.
  - If your types are defined as reverse delta and compress=N, then use the symbolic &C1BASELIB in processors to reference the type base library.
  - Use the processor parameter ALLOC=PMAP | LMAP for building concatenations using the Processor Map Allocation feature.

    More Information:

    For more information about Processor Map Allocation, see Use Symbolic Variables in Data Set Names. (see page 21)

Business Value:

These best practices help to reduce administrative maintenance.

ESYMBOLS Table

Use symbols in the ESYMBOLS table to represent common values that are used in processors. The ESYMBOLS table can hold these types of symbols in addition to the C1 CA Endevor SCM symbols.

Business Value

Using symbols to define common values once for the site makes it easier when updates are required. Also, these symbols make it easier to write processors.

Packages

The following best practices apply to packages:

- Set naming standards and use the target stage as the name prefix, if the package is *not* a promotion package.

  Business Value:

  Lists of package names are automatically sorted by target stage, which makes it easier to find a specific package.

- Use promotion packages for packages that contain Move actions only. The Promotion Package feature lets you use the same package for as many iterations and movements as needed, rather than requiring that you create new packages. If a package is not set up as a promotion package, every time it moves from one stage in the lifecycle to the next, an entirely new package must be created.

  Business Value:

  If you use a package for move processing only and you need to promote the package all the way through to production, the Promotion Package feature simplifies this process.

  Promotion packages provide for greater project control and simplify the movement of packages through the lifecycle. This feature helps you meet regulatory requirements, because it makes it easy to track the transitions in the development of your applications. Promotion packages provide auditors with a complete trail of who did what, when, and where for each iteration of the package. In addition to facilitating project control, promotion packages also make it easier to comply with a standard package naming convention.

  Additional Considerations:

  Promotion packages can contain Move actions only; no other CA Endevor SCM actions are allowed. Promotion packages keep the same package ID for the entire migration cycle.

- Enable the C1DEFLTS STOP@STG feature for promotion packages that require package ship support at a stage prior to the last stage of your software life cycle.

  Business Value:

  This best practice lets you use promotion packages, but enables you to stop them at one or more designated stages for shipment, prior to completing the promotion to the end of the lifecycle map.

- Use external security groups (external security profiles) for approver groups.

  Business Value:

  This best practice puts the responsibility for maintaining security in the security department, rather than with the change management administrators. This allows for more flexibility in maintaining approver groups by allowing more than 16 approver IDs and the addition or deletion of approvers as needed for a specific package.

- Make packages sharable by default.

  Business Value:

  A sharable package can be maintained by more than one person. If the creator of a package is unavailable when action needs to be taken with that package, having it sharable means that someone else can step in and maintain the package.

- Clean up backout members at specific intervals using the COMMIT action.

  Business Value:

  Backout enabled packages leave encrypted members in the output libraries. The COMMIT action removes those members, thus keeping the library clean and minimizing the space requirements.

- Archive after *n* days.

  Business Value:

  The Package Master file is a VSAM file which can grow quite large if not maintained. Archiving completed packages that have been committed helps to minimize that growth.

- Delete after *n* days.

  Business Value:

  The Package Master file is a VSAM file that can grow quite large if not maintained. Deleting unused packages (those that have not been accessed in a set number of days and which are still in the “Being Created” state) will help to minimize the growth of the VSAM file.

Note: For more information about archiving completed packages and deleting unused packages, see the following technical document on http://ca.com/support: TEC348243 *When was the last time your site did package clean-up? Does you site have packages that are no longer needed? Read this article to find out how ...*

Other Best Practices

The following best practices make CA Endevor SCM easier to use and let you take advantage of advanced features.

Customize ISPF Session Parameters

Set session parameters to values that help your specific development process, instead of leaving the parameters set to the default values in the delivered software.

We recommend the following best practices.

- Regularly review the ENDICNFG table, which contains the values that initiate the ISPF panels.

  Business Value:

  Most of the common session parameters that can be set are in the ENDICNFG table. While these values can be set at installation time, they should be reviewed occasionally to ensure that they still reflect your site's needs. Each of these values results in a default value being displayed when the developer enters a panel that has a corresponding field. Whereas the developer can typically change the value as necessary, most developers will want the default value to correspond to the process that makes the most sense for their site.

  Additional Considerations:

  We recommend setting the following ENDICNFG parameters:

  INTERCEPT_ISPF_RETURN

  Set this parameter to Y to ensure that the Return command (typically PF4) will not exit CA Endevor SCM, but will return a user to the CA Endevor SCM Primary Options panel, thus facilitating the user's ability to navigate within CA Endevor SCM.

  SHARABLE_PACKAGE

  Set this parameter to Y so that all packages, by default, will be created sharable. This ensures that future maintenance against the packages can be conducted when it needs to be done.

  DELETE_AFTER_TRANSFER

  Set this parameter to N so that you have a backup in case there is an issue with the transfer process.

- Set the SOFETCH parameter to Y in the C1DEFLTS table.

  Business Value:

  This parameter indicates whether an element that you have caused to be fetched should be signed out to you, if it is not already signed out to someone else.This parameter will come into play with Add (Fetch), Generate (Copyback), Move (Fetch), Transfer (Fetch), Search and Replace (Fetch), and QuickEdit.

- Instruct your developers to modify their personal defaults settings on the User Defaults panel. Specifically, each developer should change the User Defaults panel field DISPLAY MSGS WHEN RC GE to a value greater than 4.

Business Value:

The user defaults should reflect the needs of the individual user. As delivered, the DISPLAY MSGS WHEN RC GE value is set to 0, which causes CA Endevor SCM to display messages for every action performed by the developer. Setting the value greater than 4 causes CA Endevor SCM to restrict the display of messages until an actual error is encountered, so the developer will not be distracted by informational messages.

More Information:

For more information see Set User Preferences in the *User Guide*.

Business Value:

The ISPF session parameters should correspond to the processes that make the most sense for your site to make development more efficient and save programmer's time.

Review Optional Features Table

Review the Optional Features table, ENCOPTBL, with every release and service pack to check for new or changed options.

Business Value:

The table provides you with a simple, easy-to-use mechanism to customize the product for use at your site. This facility enables you to configure your implementation of CA Endevor SCM by specifying which features you want to activate. Checking the ENCOPTBL table will help keep you up-to-date with the latest optional features and how they can affect your implementation.

Additional Considerations:

The source member ENCOPTBL is found in the installation TABLES file. We recommend activating the following options:

IGNORE_SKIP_ALLOC=ON

This option enables data set name place holders within processor concatenations to assist with processor execution. It lets CA Endevor SCM ignore (bypass the allocation of) data set slots in processor data set concatenations.

If this option is activated and you specify IGNORE as the first six characters of a data set name within a processor, when CA Endevor SCM executes the processor, it examines the first six characters and, if it finds IGNORE, it does not allocate the data set for execution in the processor. In effect, CA Endevor SCM ignores the data set.

This feature was developed to address the need for empty or null data sets within a processor, which act as place holders for valid libraries, depending on the stage at which the processor is being executed without requiring the processor to allocate a null data set.

PKG_ELEMENT_LOCK=ON

This option causes the package processing routines to lock the elements that are contained in a package when the package is cast. This prevents elements from being changed while the package is awaiting approval.

DEFAULT_REVERSE=ON

This option causes the type definition defaults to change from Forward/encrypted base to Reverse/unencrypted for the element as well as the component list.

ELMNM_PROMPT=ON

This option causes an ISPF confirmation panel to display when the element name is omitted and Y is specified for the search map list option. Significant delays can occur building an element selection list for all elements across multiple environments.

INTRDR_ALTID=ON

This option overrides the default logic that forces any processor step containing an INTRDR specification to run under the USERID. If this option is set on, then the processor step runs under the alternate ID. This ensures that jobs submitted to the internal reader run under the security context of the alternate ID.

Important! This option is ignored if ALTID=N is specified on the exec card of the processor step. In this case, the step runs under the USERID.

PKGID_PROMPT=ON

This option causes an ISPF confirmation panel to display when the package ID is omitted from the package panel. Omission of the package ID can cause significant delays in the construction of the package selection list.

RESERVED_OTHERDSN=ON

This option reserves the other dataset name field in each user's profile for the C1SD4000 (footprint display), C1SF1000 (add/update) and C1SF2000 (retrieve) panels after each use. This option issues the ISPF command to save the panel field in the user’s profile, resolving to the last entry upon each display of the panel.

Use Change Control Identifiers

Use Change Control Identifiers (CCIDs) to group and manipulate similar kinds of change activity. For example, a number of different elements (COBOL copy members, COBOL programs, and so on) could be included in the same change request.

Business Value:

By tagging each change with a common CCID (such as FIX01), every change made for a change request is categorized as belonging to that CCID. Subsequently, the CCID can be used to identify and manipulate all the elements within the change request. CCIDs ensure accurate reporting, ease of package creation, and the maintenance of an audit trail.

Additional Considerations:

CCIDs can be used to integrate with the product CA Service Desk Manager.

Restrict Security for Control Files and HLQs

Restrict access to read only for the data set high level qualifiers used by CA Endevor SCM and to the files owned by the product, and allow only the CA Endevor SCM Alternate ID, CA Endevor SCM administrators and systems programmers to have higher access levels. These include the Master Control File, Package Master, and so on, as well as the base, delta, and output files.

Business Value:

Allowing read access to anyone while restricting write and update access to the CA Endevor SCM Alternate ID protects the CA Endevor SCM owned content while enabling all work activity to proceed without limitations.

Chapter 3: Performance and Maintenance Best Practices

The chapter recommends best practices to speed performance and maintain system integrity.

This section contains the following topics:

Run Full Backups and Incremental Unloads

Run full volume backups periodically and use the CA Endevor SCM Unload/Reload utility to do incremental unloads between full volume backups. Consider performing full volume backups weekly and incremental unloads daily. In addition, data validation should be performed daily.

Business Value:

Full volume backups run more quickly than CA Endevor SCM full unloads. For disaster recovery, you could restore from the full volume backup, and then add data from the incremental unload.

Additional Considerations:

The Unload/Reload utility (program C1BM5000) is a backup, recovery, and file validation mechanism for CA Endevor SCM VSAM control files (Master Control File, package data sets) and their related base and delta libraries. It allows users to backup (unload), restore (reload), and/or validate the integrity of one or more CA Endevor SCM environments in the event of a physical device failure or site disaster.

The Unload/Reload utility creates a file from which individual elements can be retrieved should one be accidently deleted. It also ensures that there is no loss of synchronization among the CA Endevor SCM files, which can occur during the full volume backups.

Use VSAM Record Level Sharing

If your site has more than 50,000 elements, use the IBM z/OS VSAM Record Level Sharing (RLS) facility to manage your CA Endevor SCM VSAM data sets to improve VSAM performance. VSAM RLS should manage all Master Control Files (MCFs), Element catalog and EINDEX, and the package data set.

Business Value:

With CA Endevor SCM making extensive use of VSAM to maintain the control information, it can, in larger installations, incur some performance bottlenecks mainly caused by VSAM open and close processing. Using VSAM RLS can help improve overall performance.

Activate Concurrent Action Processing

Activate the Concurrent Action Processing feature to speed processing of large numbers of elements. Use categories within the Global Type Sequencing member to enable Concurrent Action Processing (CAP) to execute multiple types simultaneously.

With the grouping capability, the application continues to dispatch element actions from the sequenced type chain as long as the group category value matches the previous request. Available STCs will only have to wait if sequenced type requests from one group category value are still processing and there are no non-type sequenced requests to process.

Business Value:

If you have bulk moves of large numbers of elements in a package or batch job, it can take a long time to complete the moves. Concurrent Action Processing speeds processing of batch jobs and packages, greatly reducing processing time. This feature enables CA Endevor SCM to exploit additional capacity on z/OS platforms to simultaneously execute certain element action requests, which reduces the elapsed time it takes to process multiple actions. An increase in throughput is immediate after Concurrent Action Processing has been activated.

Additional Considerations:

The CA Endevor SCM administrator must activate this feature.

Index

R

S

T

U

V