CA BEST PRACTICES

CA IT Process Automation Manager Best Practices

Recommendations for Process Design and Monitoring

THIS IS A DRAFT DOCUMENT – FEEDBACK IS WELCOME

Transforming IT Management

TITLE AND PUBLICATION DATE:

CA IT Process Automation Manager Best Practices – General Recommendations
Publication Date: April 2010
Last Update: April 1, 2010

ACKNOWLEDGEMENTS

Principal Authors and Technical Editors

George Curran
Anders Magnusson
Terry Pisauro

CA PRODUCT REFERENCES

This document references the following CA products:

- CA IT Process Automation Manager™ (CA IT PAM)
- CA Embedded Entitlements Manager (CA EEM)
- CA Enterprise Log Manager
- CA Software Change Manager

FEEDBACK

This is draft document and feedback is encouraged. Please email us at impcdfeedback@ca.com to share your feedback on this publication. Please include the title of this publication in the subject of your email response. For technical assistance with a CA product, please contact CA Technical Support at http://ca.com/support. For assistance with support specific to Japanese operating systems, please contact CA at http://www.casupport.jp.

Contents

Chapter 1: Introduction

CA IT Process Automation Manager (CA IT PAM) provides a centralized and structured approach to operations management by enabling you to define, build, orchestrate, manage, and report on automated processes spanning across different teams and roles in your organization. By automating routine administrative tasks, CA IT PAM improves operational efficiency and incident response handling, and ensures best practice and regulatory controls compliance.

This document is one in a series of papers providing best practices for making the most of your CA IT PAM implementation. The focus of this paper is on general best practices when using CA IT PAM. Additional best practice documents are available and include topics such as Maintaining Availability, Security Considerations and Performance Optimization.

Chapter 2: Best Practices for using CA IT PAM

CA IT PAM provides a very flexible set of options for designing process automation. Understanding the differences between some of those options and their impact on processing, as well as establishing a firm set of standards for their usage can significantly improve your results. This chapter provides general best practice recommendations for using CA IT PAM – both in designing your processes and monitoring them.

Note: Much of this information can also be found in greater detail in the CA IT PAM Development Guide and Programming Reference Guide.

Best Practices for Process Design

In CA IT PAM, processes are defined by Process objects that define a sequence of linked steps. These steps are defined by placing Operators, representing various actions performed by CA IT PAM modules, in the Process. Logical and control operators are used to define starting\stopping points, branching and iterations within a Process.

Designing a Process requires careful and thorough analysis of the tasks that are being automated. Following is a list of guidelines we will look at in the next sections:

- Establish and Follow Meaningful Naming Conventions
- Take Care when Renaming Objects
- Working with Data Sets
- Working with Resources
- Mirror, Mirror
- Using Calendar and Agendas
- Exporting and Importing Modules and Packages
- Managing Change Conflict
- Don't forget to Check in your Changes
- Understanding Stops and Operators
- Use Roles to Manage Access
- Re-use and Reduce CA IT PAM Objects
- And Recycle / Purge Deleted Objects to Avoid Name Collision
- Detached Mode
- Know your Exits Ahead of Time
- Use the Javascript Library
- Mind your Expressions
- Use Comments
- Always use Exception Handlers
- Test, test and test again
- Use the User Request Form

Establish and Follow Meaningful Naming Conventions

It is important to establish and consistently adhere to a standard naming convention. This will both improve the efficiency of the design process and assist in troubleshooting, if needed. You can use naming conventions to visually connect specific types of processes, associated departments or roles or other characteristics.

Naming conventions can also be used for user roles.

Some important rules for the naming convention policies are:

- Always treat Datasets as case sensitive – but do NOT assume that they are.
- Do not use the same name for a Process Dataset variable and an operator in the same Process.
- Avoid including blank spaces in object names. Typically, alphanumeric characters and underscores (_) are allowed in object names. A list of the allowed characters for each operator type can be found in Appendix A of the *Administration Guide*.

Following are some additional best practices for the naming conventions policies:

- Use a prefix, for example "r_", to indicate processes which can be re-started and other processes that you would like to be able to single out or sort in a specific way.
- Use names that describes an object's function rather than where it is executed. This will minimize your work when/if you move a process, Touchpoint or other IT PAM objects to a new updated server.

Take Care when Renaming Objects

Renaming objects that are used in production should be avoided whenever possible. If you need to rename an object for any reason it is critical that you carefully analyze how this change could affect your processes. The reason for this is that objects are typically referenced by name and, if you change the name of an object (for example, a Dataset, an operator or a Touchpoint), any process that references this object will no longer be able to identify it. As a result, the process will no longer work as designed.

Working with Data Sets

In CA IT PAM, a Dataset object is used to store information that can be shared among instances of the same process or between different processes. The information defined in a Dataset can be modified manually by users or administrators or automatically during the execution of a Process. Within a Dataset, you can define pages that are used to visually organize the various parameters or fields that the Dataset will contain. It is important to understand the differences in scope between the five types of Dataset objects:

- **System Datasets** contains predefined, read-only variables that are available in context of the entire IT PAM domain. These variables access system parameters and are made available by the System keyword. The scope is global to all operations in any IT PAM environment – and it cannot be edited.

- **Process Datasets** define variables in the context of a Process and are included in every instance of a Process object. Within the context of a particular Process, this is also called the "local Dataset'". A Process Dataset has a scope that is accessible, as well as modifiable, from all operators within the Process and can be initialized from the parent process or from external applications. The final value is readable from the parent process.
- **Operator Datasets** define variables in the context of an operation within a Process or Agenda. They are included in every instance of an Operator that is added to a Process or Agenda object. An Operator dataset has a local scope in that it can be accessed and modified within a Process.
- **State Policy Datasets** define variables in the context of a State Policy object. A State Policy Dataset is referred to by expressions within the State Policy as the Process Dataset for the policy and is primarily accessible to Action scripts in its Process.
- **Named Datasets** are distinct automation objects that are edited and saved to a folder in an IT PAM Automation Library. A Named Dataset is accessible to all processes in the Library in which it is contained. It is defined by a Dataset object and has Library-level scope.

When and how to use Datasets

Datasets is an excellent method to pass and/or share information between different types of objects. But it is important to be aware of the scope of the information, (see details in the list above), and it is considered a best practice to store information as local as possible without causing unnecessary duplication of data.

Operators that return information that is more complex than a binary response typically use the operator dataset. If you need parts of this information to be accessible to other processes you need to decide who needs to see this information (a "Named Dataset" is available for the complete environment, while a Process Dataset is only available in the process itself and its parent process) and what parts of the data need to be made available.

A few examples are:

- Data that is frequently shared among multiple processes and operators should typically be stored in a named dataset. A typical example of this is usernames, passwords, database and server names for certain applications. Storing these in a named dataset greatly simplifies the process when any of these parameters needs to be updated. An update can include anything from a simple change of password to a situation where you have moved an application server to a new renamed server.
- When storing usernames, passwords, server names or other parameters in a shared dataset be sure to use different variables for different logical functions. For example:
    - > If Application A and Application B are both hosted on Server X, their Application Server Name should not be stored in the same dataset variable. The reason for this is that you might, at a later point, decide to re-architecture the solution and separate these onto separate servers.
    - > If Application A and Application B both use "root" to connect to a specific function, store this username in application specific variables. This will later give you better flexibility in the event you (or your security team) need to have separate user IDs for the individual applications.

- If you, from a parent process, would like to use the data from an operator inside a child process the data should first be moved from the Operator Dataset to the process dataset for the child process.

If you are creating custom script operation on a Windows or a Unix system, it is also important to be aware that you can easily pass data from the script and back into the operatorss dataset. Details on how to do this are provided in the "Read Operating System Values into IT PAM Dataset Variables" section in the *CA IT PAM Development Guide*.

Referencing Datasets

Depending on where in the "scope" you are and the type of datasets you are referencing there are slightly different methods to address the datasets. Note that keywords are highlighted using a bold font while other names are referencing variable names.

- **Access Operator Dataset variable from inside the process where it resides**

  There are a few supported syntax options to accomplish this:

  `Local.Operator_name.field_name`

  or

  `Process[Operator_name_expression].field_name`

  or

  `Process[Operator_name_expression][field_name_expression]`

- **Access a Process Dataset**

  To access a process dataset use the following syntax:

  `Process.field_name`

  or

  `Process[expression]`

- **Access the parent's Process Dataset**

  Using the following syntax you can directly access the dataset of the parent process (or parent state policy):

  `Caller.field_name`

- **Access Named Datasets in an IT PAM library**

  To access a dataset in a named dataset use the following syntax:

  `Datasets[dataset_path].field_name`

  Or

  `Datasets[dataset_path][field_name_expression]`

  For example

  `Datasets["/Demo/data/default"].PAM_DomainServer`

  will return the value of the variable "PAM_DomainServer" located in a dataset called "default" which is located in the /Demo/data" library path .

- **Access System Datasets**

  CA IT PAM contains a number of predefined variables that provide system information. The available variables are:

```
DATE, DAY, FIRSTDAYMONTH, FIRSTDAYNEXTMONTH, FIRSTDAYPREVMONTH, HOST, LASTDAYMONTH,
LASTDAYNEXTMONTH, LASTDAYPREVMONTH, MONTH, TIME, TIMES, TOMORROW, WEEK, YEAR and YESTERDAY.
```

See the *CA IT PAM Development Guide* for more information about these.

To access a System Dataset use the following syntax:

```
System.field_name
```

Or

```
System[field_name_expression]
```

In addition to these direct references to the dataset there are a number of other keywords and properties that are useful when accessing datasets. A few common examples of this are:

- **length / size**

  To determine the length of an indexed field, CA IT PAM supports both the JavaScript length property as well as the CA IT PAM proprietary size. Both options work the same way with the exception that "length" is a read-only property while "size" also can be used to change the number of elements in an indexed field. The following example uses size, but it is interchangeable with length.

  ```
  dataset_reference.indexed_field_name.size
  dataset_reference[indexed_field_name_expression].size
  ```

- **OpName**

  The pre-execution and post-execution code for an Operator can access the name of the current Operator using the OpName keyword. Use the following syntax to specify an Operator Dataset variable in the execution or post-execution code of that same Operator:

  ```
  Process[OpName].field_name
  ```

  In additon the OpName keyword can be used directly to store the operator name for future use:

  ```
  Process.iName = OpName
  ```

Looping through indexed elements of a dataset field

To loop through all the elements of an indexed Dataset field, first use an Interpreter Module Calculation operation to initialize the CurrentIndex element for the Dataset field to 0. Then, in the operation properties of the operation you want to loop, use the Size element setting for the indexed field as the Repeat count value on the Loop tab (for example, Local.X.Size). To increment the CurrentIndex setting after completing each iteration of the loop, use an expression on the Post-Execution Actions tab of the operation properties (for example, Local.X.CurrentIndex=Local.X.CurrentIndex+1;). In this case, the CurrentIndex element is the counter for the loop. Note that the IT PAM Dataset field is a zero-based array. The first element of an indexed Dataset field is indexed by 0 and the last element is indexed by one less than the value of the Size element for the field.

Inside the loop you can use the CurrentIndex setting to access elements of the indexed field in calculated expressions (for example, Local.X[Local.X.CurrentIndex]).

See the "Iterative Statements" section in the *CA IT PAM Development Guide* for additional options and details.

Working with Resources

Resources are used to represent, and control access to, a particular entity within your IT environment, such as a database, a log file that will be updated by multiple processes, or access to an application that uses CAL-based licensing. A single resource object may contain multiple parameters representing various entities. For each parameter you can specify the total number of units associated with the parameter as well as the current number available and free. You can also lock and unlock access to a parameter regardless of the number of units free.

Resource object are used to synchronize independent processes that rely on common elements of the infrastructure. You can use them to meter systems resources to IT PAM Processes that are required by mission critical tasks on your system. For example, you can limit the number of simultaneous FTP connects used by CA IT PAM. You can also use resources to order operations by allowing a successor process to start only after an antecedent process releases a resource.

Resources can be used to:

- Balance the processing load across all Processes running on a particular Touchpoint
- Synchronize the execution of Processes that cannot execute in parallel
- Implement environmental level locks that simultaneously enable or disable multiple processes

A Resource consists of a maximum number of units, the current value of available units and a flag indicating whether the resource is locked. A Resource operation can consume or free any specified number of resources required to allow a developer or administrator to fully tune the load balancing on a particular Touchpoint. It can also lock a Resource to prevent consumption of Resource units by any other process. The maximum number of units of any resource is an arbitrary value that you can calibrate and fine tune for your particular system requirements. The number of resources used by any Process is also arbitrary. You are free to allocate Resource units to IT PAM Process as best suits your implementation requirements, however, keep in mind the following constraints:

- The currently used value of resource units is always smaller than or equal to the maximum value of the resource
- A resource-dependent process must wait until its specified number of units are available
- Operations cannot consume units from or lock a resource that is locked by another process as long as that resource is locked

Lock vs. Pool

Resources can be modeled in one of two ways: a Lock or a Pool. For example, if you are dealing with product licenses, which are limited in number and you choose Resource to synchronize execution your model would be a pool. In this case you could use the Take and Free Resource Primitives options.

If, however, you are waiting for an execution of a particular path of a Process to complete, then the model is a Lock. In this case, you would use the Lock and Unlock Primitives of the Resource.

There are also scenarios in which both uses of the same resource are useful. For example, locking a resource that is normally used as a "pool" allows some processes to suspend execution of processes already waiting for resources to be used (e.g., for maintenance windows).

When you use Resources be aware of indefinite waits and deadlocks. To avoid these, write a resource release process that can be used in the event your only recourse is to reset the resource to its original good state.

Mirror, Mirror

Both Agents and Orchestrators regularly check for changes to user uploaded files in the Domain repository (.c20repository). In addition, Orchestrators also check for changes to system files in this repository. If one or more files have been changed the Orchestrator or Agent will download the new version of the file(s) and save it in the local repository.

The mirroring interval for an Orchestrator can be defined in the "mirroring" tab in the CA IT PAM Clients Configuration Browser. The mirroring interval for Agents can be set in the individual agents properties tab (in the right hand pane when you select the Agent you would like to modify in the Configuration Browsers Agents panel).

Using Calendar and Agendas

Process scheduling is managed through Calendar objects and Agenda objects. Calendar objects are used to define date conditions under which tasks (Processes or individual operations) can be started. Agenda Policies are used to schedule tasks using the Calendar object and/or explicit dates.

Date-Time Operators define time and Calendar constraints for executing branches in a Process. For example you can conditionally execute or delay branches in a Process depending on whether a specified date and time has passed. Operators in an Agenda are started according to specified Calendar and time conditions.

If there is a conflict (for example, if a day falls in both the include and exclude buckets), an Operator will not be scheduled on that day. You can use the following options to define behavior when there is a conflict:

- **Avoid**: Specifies the Calendar object that defines days to exclude for scheduling (those on which an Operator may not be scheduled).
- **Delta**: Specifies the number of calendar days an included day is shifted when it conflicts with an excluded day. The shift depends on the whether the value is positive, negative, or zero. A negative value shifts forward (earlier), and a positive value shifts backward (later). When this value is zero, the include day is simply skipped without rescheduling the task. For example, an Operator is scheduled to execute every Friday. If a specific Friday is on the excluded list, the Operator will not run on that Friday. However, if the Delta is set to -1, the Operator will run on Thursday (one day earlier) instead, as long as Thursday is not on the excluded list. Similarly, if the Delta is set to 2, the Operator will run on Sunday (two days later) as long as Sunday is not on the exclude list.
- **Open days:** Only counts days that are not on the excluded list when shifting to avoid a schedule conflict. For example, an Operator is scheduled to execute every Friday. Weekends are on the excluded list. If a specific Friday is also on the excluded list, the Operator will not run on that Friday. However, if the Delta is set to 1, the Operator will run on the following Monday (which is one day later, not counting the weekend, which is on the excluded list) as long as that Monday is not on the excluded list.

- **Maximum shifts**: If the Open days check box is not selected, a day that is shifted may fall on another excluded day. This option defines the maximum number of shifts that are allowed if repeated shifts fall on excluded days. For example, Operator A is scheduled to execute every Wednesday and Operator B is scheduled to execute every Friday. For both Operators, the Delta is set to 1, Maximum shifts is set to 3 and weekends are on the excluded list. If a specific Wednesday is also on the excluded list, then Operator A will be shifted by 1 calendar day and will run on Thursday. If a specific Friday is on the excluded list, then Operator B will be shifted by 1 calendar day for each shift, up to 3 shifts total. Since the weekends are excluded, Operator B will be shifted the maximum of 3 times and will run on the following Monday. If that Monday is also on the excluded list for some reason, then the Operator will not be rescheduled since the Maximum shifts has been reached.

When a Calendar is not specified, all days are eligible unless the **Only consider manually scheduled dates** option is selected. If this option is selected the Operator is executed only on those days that are manually scheduled in the Dates Added list.

Working with Environments / Libraries

When working with multiple parallel environments it is important to establish a well defined, standardized library structure for your IT PAM objects. This highly simplifies the troubleshooting process for any administrators that are working across multiple environments.

The recommended method of sharing CA IT PAM objects between Environments is to use Automation packages. Following this practice will partially enforce the usage of the same library structure since all objects that are imported using a package will use the absolute path that was used in the parent Environment.

IMPORTANT: Since all other Environments will inherit the library structure from the Environment in which the objects were packaged it is critical that the naming standards take this into account. This is especially important on the development environment.

Exporting and Importing Modules and Packages

A CA IT PAM "Package" is a logical group of IT PAM objects that can be used to package changes, fixes or enhancements as a controlled entity. Packages are also typically used to migrate CA IT PAM objects between Environments.

Creating CA IT PAM Packages

Any CA IT PAM object can be added to a package and, unless otherwise specified, the current version of the document will be included in the package. To avoid unintentionally including a previous, or incorrect, version of an object in the package it is considered a best practice to first check in all objects before they are imported into an automation package.

In most ways a package should be treated in the same way as other CA IT PAM objects. This includes applying good naming standards that indicate the purpose of the package. A package can also be controlled by versioning, and access to it controlled through ACLs. As with other CA IT PAM objects, you can also provide a description for it in the "General" tab if you open its properties. For any objects that are considered "complex", such as more complex processes, should have a description that describes the content and purpose of the package.

IMPORTANT: All Environments using a particular package will inherit the library structure from the Environment in which the objects were packaged. Therefore, it is critical that the naming standards take this into account. Note that this is especially important in a development Environment – which is where packages typically originate.

The recommended steps to create a Package are:

1. Open the CA IT PAM Client and navigate to the Library Browser where your objects are located.
2. Ensure that all objects that will be included in the package are checked in.

   (Or that the version that you are planning to package is set as the current version.)
3. In the left hand pane navigate to the folder where you would like to place the package
4. Right click on the folder and select "New Object / Package"
5. Rename the newly created package, using your naming standard provide a descriptive name.
6. Right click on the package, select properties and update the Description field so that it describes the content and purpose of the package. Click OK
7. Open the Package by double-clicking on it.
8. Click on the green ➕ sign in the menu to open the Object Browser and select the object you would like to include in the package. Click OK
9. If you are planning to use another version than the "current" select the appropriate version.
10. Repeat step 8 and 9 for each object you would like to include in the package.
11. Click on the Save Icon for the File/Save menu.
12. Check In the package by clicking on the Check In icon and click OK

Migrating CA IT PAM packages

CA IT PAM Packages can be exported from one library (Environment) and later imported into the same or other environment. To simplify migration of packages between your various Environments it is recommended that you use a standardized directory structure across all your Environments (see "Working with Environments / Libraries").

When you import a package, the package itself can be imported into any location in the library. When the package is imported all CA IT PAM objects within the package will also be imported in their original directory structure. Note, even though the package itself can be imported with a relative path, all of its content will be imported using absolute (root-level) paths. If any of the directories with IT PAM objects do not already exist they will be created during the import. An updated CA IT PAM package can be imported into an environment prior to the release date and will not be active while importing them:

- Ensure that the Imported version is not set as the Current Version
- Do not make the Custom Operators available

The exception to this is if the CA IT PAM objects do not already exist in this Environment. In that case this will be the only active version of the object.

Note: Since each CA IT PAM Environment will keep track of the version number for its own objects, there may be some disparity in version numbers between different CA IT PAM Environments. Because of this, they should be tracked using package name and Descriptions.

Also remember that, depending on your implementation, any Environment-specific configuration might need to be reapplied if it has been overwritten by the imported package.

Rolling Back to Previously Deployed CA IT PAM Packages

If, for any reason, you need to roll back a deployed package, the individual CA IT PAM objects that make up the package need to independently be rolled back manually. This is can be done by setting the Current Version of each CA IT PAM object to the previous version, however this is a somewhat tedious task. Therefore great care should be taken to ensure that changes have been thoroughly tested and validated during QA.

As an alternative to actually rolling back the CA IT PAM objects, you can redeploy the previous version of the CA IT PAM package. This is a relatively easy procedure if you have previously followed strict change procedures and if no major changes have been made to the production version.

Managing CA IT PAM Packages

Any change request for new, updates or fixes of CA IT PAM objects should be distributed as a Package, if packaged correctly you can guarantee that all required objects are delivered as one entity and that they are compatible with each other. If necessary, this also simplifies the process of managing any updated CA IT PAM packages in an external Configuration Management System such as CA Software Change Manager.

To ensure a secure and safe production environment it is recommended that your development include a strict change management process. A suggested process is outlined below:

- **Development**
    - Create Processes
    - Development Testing
    - Package for Deployment (Create CA IT PAM Package, Export as XML)
    - Optionally check in into Configuration Management Tool
- **Software Configuration Manager (optional, but recommended)**
    - Code Review
    - Send to Testing
    - Promote/Demote the Configuration Management Tool Package
    - Manage Packages using CM best-practices
    - Prepare process for Implementation
- **Quality Assurance**
    - Import the Package into CA IT PAM Library
    - Make specified configuration changes (Passwords, Dataset information, Security Requirements, other variable configurations)
    - Validate/Verify the Functional Implementation of the Package
    - Demote to Development in case of problems, issues

- **Production Test/ Staging**
  - Import the Package into CA IT PAM Library (in environment that closely mirror production)
  - Make specified configuration changes (Passwords, Dataset information, Security Requirements, other variable configurations)
  - Validate/Verify the Functional Implementation of the Package
  - Optionally, Test DR scenarios, Load Testing, etc.
  - Demote to Development if necessary
- **Production**
  - Import the Package into CA IT PAM Library
  - Make specified configuration changes (Passwords, Dataset information, Security Requirements, other variable configurations)
  - Validate/Verify the Functional Implementation of the Package
  - Demote to Development if necessary
  - **DEPLOY**

Summary – Final thoughts on Working with Packages

A few important points to remember when working with Packages are:

- All Fixes and changes should be well documented and originate from Development Environment
- Packages should be named appropriately for version tracking and auditing of Changes, Fixes, Updates, Upgrades, etc.
- External Configuration Management Tool can be used to provide additional auditing and change tracking capabilities
- ACLs should be set appropriately (on groups and folders) to manage access rights on CA IT PAM Objects with minimal issues
- Naming standards should be applied in a consistent way between CA IT PAM Environments to facilitate minimal changes to the individual CA IT PAM Objects. This include using the same Touchpoint names, library structure etc
- All Run-time Environment specific changes should be captured in CA IT PAM Datasets for manageability

Managing Change Conflict

A change conflict can occur if the Development team has check out an object for enhancements (such as long term changes) while the Operation team needs to check out the same object to quickly address a more specific issue.

This can be addressed in a controlled way by using the following procedure:

1. The Operation team talks to the Development team to coordinate the handover.
2. The Development team checks in their object its current state.

   This ensures that no changes are overwritten or lost. Since the changes are not ready for QA or Production it is important "Set Current Version" is not checked.

   Note that Current Version now has a lower number than the Latest Version. For example, while the current version number may be 5, the latest version number – the on just checked in by Development - may be 6.
3. At this point, Development can hand over ownership of the object to the Operations team which can then check out the *current* version.

   Since the Operations team is checking out the current version (version 5, in our example) this will not include the changes that development has been working on.
4. The Operations team can edit this newly checked out object and include their quick fix
5. After the fix have been tested the Operations team checks in the fixed version. Since this version that should be used they can check "Set Current Version"

   In our example, the current version is now version 7, while the development version - with the long term changes – is still version 6.
6. The Operations team can now hand the object over to the Development team, along with detailed instructions on what they have changed.
7. The Development team checks out the version they were working on previously (version 6 in our example) and continue with their more long term work. However, they will also need to review the changes that were made by the Operations team in order to determine if these changes should be merged into the Development version. If that is necessary, it will need to be done manually.

Don't forget to Check in your Changes

All automation objects in CA IT PAM are associated with version control. Therefore, if you create an automation project, such as a process, resource or data set, make sure you check it in. Otherwise, any changes you intended to make to that object will not be available to other objects. If you make changes to an existing automation object, you need to check it out first, prior to making those modifications.

Understanding Stops and Operators

In CA IT PAM, execution happens in parallel in all parallel paths and processes by default. As a result, whenever a Stop is encountered in one of those paths, execution stops. The **Stop** operator terminates a Process and determines the return value and can be configured as either an **Abnormal Stop** or a **Normal Stop**. When a Run Process Module runs a Process in attached mode, the **Normal Stop** sets the Operator Dataset Result variable for the Run Workflow Operator to 1, while the **Abnormal Stop** sets the Result variable for the Run Workflow Operator to -1.

If you are trying to synchronize the parallel paths of the *same process* consider using the AND operator. If you are trying to synchronize across *multiple processes*, consider using the Resource option.

The **And** operator defines a synchronization point between all entry links to it. Exit links from an **And** operator are activated only after all the entry links to it have been activated. Use an **And** operator to synchronize multiple branches of a process with a logical **And** condition when two or more separate branches of a flow must all be completed before beginning one or more additional branches.

The **Or** operator also defines a synchronization point between all entry links to it. Exit links from an **Or** operator are activated after at least one of any number of entry links to the operator has been activated. At least one of two or more separate branches of a flow leading to an **Or** operator must be completed before beginning one or more exit branches.

Use Roles to Manage Access

Best practice is to define profiles based on role usage as opposed to assigning permissions based on individual user IDs. Advantages to this approach include:

- Saves time and money since you only need to configure the detailed security profile for a role once – rather than individually setting one for each user.
- Enables you to quickly and correctly configure access for new employees or existing employees who have changed roles within the organization.
- Enables you to securely manage access for temporary workers that need access to the system for a short period of time.
- Minimizes delay and frustration due to incorrect security settings; if a role needs access to additional resources you can quickly change this for everybody in this role with one change.

Managing access rights for individual users can quickly become nearly impossible to manage, and even worse, if everybody accessing the system is using the same user ID it will be impossible to trace “who did what”. Furthermore, any changes to the personal settings will immediately affect everybody and therefore cause a support nightmare where people no longer recognize their environment.

Re-use and Reduce CA IT PAM Objects

Modularity is a powerful concept in development. Therefore, whenever possible, divide any large end-to-end Processes into logical, self-contained sub-Processes, also known as “child Processes.” Consider defining subprocesses for any activity that you are likely to invoke in different situations. For example, notification\escalation processes, opening help desk tickets, etc. Two main criteria for determining when to create a sub-Process are:

- Is that sub-Process used in a large number of other processes in your IT organization?
- Is that sub-Process repeated a number of times as part of an overall Process?

If the answer is "no" to both questions, then it probably does not make sense to create the sub-Process. If the answer is "yes" to one or both, then it may make sense. Some examples of where a sub-Process could make sense include:

- Steps that belong completely to one IT silo
- Steps that interact with one tier of an application or system

Once you have decided to create a sub-Process, you will probably want to make it available so that others in your organization can use that sub-Process instead of having to reinvent the wheel. In order to promote this and make the use of the sub-Process successful, you will need to define inputs and outputs of the sub-Process to create a clean hand-off, just like you would with manual processes. One example of processes that often make sense to define as sub processes are processes that handle escalations and notification. These processes will likely be called by numerous other processes and, therefore, it makes perfect sense to define and maintain them in one central place.

Also consider creating custom operators to extend, encapsulate and re-use functionality. Whenever you have extended a particular functionality you can make this available to others to use by creating a custom operator.

And Recycle / Purge Deleted Objects to Avoid Name Collision

When you delete an Automation Object it goes to the Recycle Bin. If you want to use the same name to create another object you need to purge the deleted object from the Recycle bin first.

Detached Mode

It is important to understand the difference between Run Process and Run Process in Detached Mode. When a sub-Process is invoked from a main Process, using the attached mode, the main Process will not be processed on that same branch until the sub-Process completes execution. Parallel branches (if any) will not be affected. This is not the case when sub-Processes are invoked in detached mode. In that case, the sub-Process is queued for immediate or later execution and the parent proceeds to the next step.

Use "Run Process in detached mode" if:

- You do not need the child Process to finish before continuing
- If you not need to take different paths based on the result of the child Process
- If you not need the child Process to return ANY information at all to the parent Process.
- If you need to start the child Process in the future

Otherwise, use "Run Process"

Know your Exits Ahead of Time

All Operators come standard with two default exit points and that is sufficient in most cases. However, in several instances, you may want to handle custom exit conditions. This allows you to automate additional behaviors. There are cases where the disposition of a possible outcome is not easily scripted and, in those situations, you can use exception handling to raise alarms or interaction forms to ask users to choose amongst several dispositions/options.

Use the Javascript Library

CA IT PAM provides a rich set of functionality through the Javascript library. See the *Programming Reference Guide* for more details. Take advantage of this functionality – consult the library! You can also define external libraries of Javascript functions and include them in code you write inside process definitions.

Mind your Expressions

When using expressions with special characters (especially the backslash "\" or the percentage "%" characters) you need to be cautious. In most cases (i.e., for most operators) it is considered a best practice to use normalized file names with forward slash "/", even when specifying a path on Microsoft Windows machines. However, in some cases (especially within custom operators) this is not supported and you might need to use the standard notation with backslash "\". When you have to use this you should typically escape it with an extra backslash. In other words, use "\\" when you normally would need a "\". For example, to refer to the "itpam\scripts" directory, your expression would look the following:

```
/itpam/scripts
```

or

```
\\itpam\\scripts
```

If you use the percent character (%) you need to substitute it with %%%% to escape the default meaning.

Another option is to make these types of escape maneuvers unnecessary by storing the strings (paths) in variables in datasets.

IMPORTANT: Since these special characters are treated slightly differently depending on how the operators are implemented and IT PAM allow custom operators to be implemented with any script language it is highly recommended that you carefully test any operation using special characters unless you are absolutely sure how what is required for this specific operator.

Use Comments

Comments are a useful way of documenting the entire process and should be utilized wisely and effectively.   Keep in mind that there may be two different audiences for your comments – other Solution Developers and Operators. Developer directed comments should depict the design and implementation details that need to be understood for the sake of maintenance and enhancements. Operator directed comments, typically include information about how to restart, manual execution and troubleshooting.

Using Touchpoints

Touchpoints map symbolic names to Orchestrators and Agents, which are then used to identify the Orchestrator Agent within an Environment. This provides a layer of abstraction between CA IT PAM and the network topology, allowing CA IT PAM operations to be configured without explicitly specifying host information. A user configuring a CA IT PAM operation selects a symbolic name from a list of all the Touchpoints in the Environment that are configured to run the module that supports the operation. This indirection allows users to substitute hosts at runtime and define multiple CA IT PAM Environments (such as development, test, and production) in which the same Touchpoints are mapped to different real hosts.

Always use a variable to refer to a Touchpoint – rather than hard-coding the name. Doing so enables you to easily swap Touchpoints without having to manually change it in a lot of places. This also allows you to run the same Process against different machines to be determined at runtime. An example of this would be a process that installs\configures a software package on the target Touchpoint. Having a Touchpoint as a variable allows it to be executed in different contexts.

You can map more than one Agent to a single Touchpoint. There are two reasons to do this.

- To provide redundancy and failover among Agents on different hosts
- To provide load balancing among Agents on different hosts

When there are multiple Agents mapped to a Touchpoint, CA IT PAM switches operations to a different Agent when an active Agent becomes unreachable. Agents mapped to a Touchpoint can be assigned different priority numbers, so Touchpoint operations run preferentially on the highest priority Agent that is currently available. To provide load balancing among equivalent hosts, you can assign the same priority to multiple Agents assigned to a Touchpoint. For example, a Touchpoint may be mapped to 20 Agents on 20 different host computers. If you assign the same priority to all of the Agents, Touchpoint operations then run randomly on any available Agent.

Another feature that can be useful when working with Touchpoints is that you can group multiple Touchpoints into one Touchpoint group. This can be useful for multiple reasons. For example, if they are logically grouped it makes it significantly easier to find the Touchpoint you need. You can also assign a process to a complete Touchpoint group and, in this way, execute the process, in parallel, multiple Touchpoints.

Note: If you execute a process on a touchpoint that include multiple agents this provides high availability or a loadbalancing functionality and the process is executed on one of these agents, if you instead execute a process on a Touchpoint group the process will in parallel be execute on all Touchpoints in this group.

Always use Exception Handlers

The Exception Handler allows you to define the actions that get kicked off when an exception occurs. For example, if you are reading information from a database and have only defined what the next Operator is when you can successfully read from the database, if the read is not successful, then the Exception Handler would be initiated. As another example, let's say that the operation based on the result from this query will take one of three paths depending on if what you read is "1", "2", or "3". This would be achieved as Custom Ports. If you unexpectedly read a "4" during a Process this would, again, cause the Exception Handler to be initiated.

Exception Handlers define actions to take when one of the following exceptions occurs

- Unidentified Response – This occurs when an Operator has no path to take after it has executed
- System Exception – Invalid Touchpoint name, system communication problems, unreachable agent
- Abort – Operator aborted by user or system

Note that an Operator must have at least one port connected to another Operator for the Exception Handler to kick in. If an Operator has no paths defined at all, CA IT PAM will assume that you do not want to continue down that path and will not initiate the Exception Handler.

Defining meaningful Exception Handlers can ensure that you are catch any "unexpected results" from the process flows.

One thing to think about is whether you want to use a Stop Operator in your Exception Handler. If you do, then the Process ends when it hits the Stop. Another option is to not use a Stop Operator. After taking the last step in the Exception Handler, the Process would go into the waiting state. At this point, the Process owner or another admin could bring up the Process and interact with it to work around the issue.

In a Production environment, when a System restart happens use Exception Handlers to notify an Event Management System.

Test, test and test again

CA IT PAM has a powerful simulation feature that can enable you to test the entire process – including remote endpoints. You can use the process simulation properties for every operation in a process – and override the process settings in individual operations. Simulation can be used during development to test dependent branches in a Process – without having to execute an operation.

Simulation can be run in either of two modes:

- Local: operation is not processed and IT PAM does not call on the associated module or check the module parameters. It does check for an application programs, validate an operation's execution Touchpoint and valid user credential for a file transfer.
- Distant: Workflow Module calls the associated module which checks parameters before returning the results but does not actually run the operation. If the parameters are incorrect, the simulated operation fails regardless of the specified outcome. If the parameters are correct, the module returns the specified result.

These modes are mutually exclusive.

There are many reasons to use the simulation feature. For example, during development, local simulation mode can be used to verify execution within a process or to confirm the synchronization of interdependent processes. Distant simulation can be used to verify the configuration and parameters of operations in a process and to verify the validity of dataset values.

Since processes can span different systems, such as help desk systems, it is always best practice to test connectivity between different systems externally first, using a test Process, before attempting to execute a CA IT PAM Process that utilizes those systems. For example, you can create a test process to test ticket creation. Doing so will enable you to catch errors quickly, before they impact actual business processing.

Limitations of Simulation

A simulated Operator does not return the dataset variables that you would expect when running normally. For example, if you are running the "Get Processes" Operator in the Windows Module, you expect to get a list of Windows processes along with a number of attributes. You may also decide to process this data as part of your Post-Execution code or to use the data as an input to a later Operator. However, in a simulated Operator, that data is not returned, so your Post-Execution code or that later Operator may not work as expected.

A good way to handle this issue is to create dummy data as part of the Pre-Execution code for the simulated option. When you no longer need to simulate, you can then remove this simulated data.

Test Considerations

If a process will be connecting to different kinds of Systems, such as Help Desk ticketing systems, you should always test connectivity to these systems externally, using a test CA IT PAM Process prior to executing any CA IT PAM Process that utilizes those connections. Doing so will help you to catch errors quickly and early.

You should also use an External Client to test an External Interface prior to incorporating it in a CA IT PAM Process. This allows you to identify and correct errors (and save time!). For example, you could use an External Client to validate the SOAP interface. This is the same approach that is typically is used during basic troubleshooting. If something goes wrong, or as in this case, you need to ensure it doesn't cause a problem, it is a good idea to first test the underlying process using an external and completely separate client. If this work you at least know that the basic infrastructure is working.

If your Process will be executing a script or program you should test those scripts\programs before incorporating them into the Process.

Validation is an important part of deploying a package. You want to make sure that all components of the package work before it is allowed to go fully operational in the production environment.

Rarely-executed mission-critical processes, such as system failovers, should be tested periodically to make sure that the processes and people stay fully current and functional in the production environment. Periodic testing prevents incidents from decaying into full-blown crises.

Use the User Request Form

Consider establishing a policy whereby users who do not use the CA IT PAM Client must use a User Request Form to invoke Processes that are intended to be invoked in an ad-hoc fashion such as a process to restart a multi-tiered application or a process to run a system check. You can define access controls on the request forms to allow users to invoke such processes. This exposes more processes to more authorized people, without requiring them to go through the full CA IT PAM client or exposing them to the internals of the process.

Processes available through request forms are also easier to handle by external applications through web services. Inputs and outputs are defined, forms can be presented to the external application users, there are calls to browse for available start requests, etc.

Monitoring CA IT PAM

CA IT PAM defines two automation objects for monitoring Processes and operations. You can graphically monitor current operations using a Process Watch object or you can monitor text traces using a LogViewer object.

In addition, you can use the JMX Console to monitor the state of the CA IT PAM installation itself.

Process Watch Monitoring

Process Watch objects are special CA IT PAM objects that enable you to view all CA IT PAM operations related to a particular category of ownership. For example, a data warehouse team would require access to a Process Watch object containing shortcuts to all extract, transform, and load (processes) for populating data warehouses.

You can use Alerts in Processes to alert operators, administrators, and others about errors or incidents, or to simply inform people that a process or task is complete. Best practices recommend creating separate Process objects that not only trigger alerts but also handle escalation and perform other tasks related to those alerts. The alert Process can then be invoked from other Processes, as needed, using the Run Process operation (see the Workflow Module section in the *CA IT PAM Programming Reference Guide* for more details).

CA IT PAM supports sound alerts, e-mail alerts, and telephone or pager alerts. You can use any or all of these depending on your particular requirements. For a condition that requires immediate operator input at a console, you can use a sound alert. For urgent error conditions, you may need to use a TAPI Alert to reach an administrator by phone or pager. For less urgent errors or reports, you can use a MAPI Alert to send e-mail messages. Different alerts can be used for different users. For example, you might have a group for managers that receive monthly reports, and an administrator who is notified after a process succeeds or fails.

Note: It is better to use e-mail as a general mode of alerts (or in addition to other modes), because message delivery is fairly well guaranteed. Telephony alerts depend on modems and external networks and hardware, which may fail to deliver a critical message.

Custom Process Watches

You can use either the Default Process Watch or a Process Watch object to monitor and edit an instance of a Process. The Default Process Watch groups various states (all, queued, running, waiting, suspended, ended, normally ended, and abnormally ended) of instances of the Process on an Orchestrator. Since a typical CA IT PAM deployment includes a number of concurrent use cases that are accomplished through several processes, datasets, resources and other Automation Objects, using the Standard Process Watch might quickly result in a sensory overload. A useful alternative to this is to create a Custom Process Watch that includes only the Datasets and Resources associated with a particular user case. Then, whenever you need to monitor the progress of a particular use case, you can simply view the custom process watch. This would enable you to monitor the status and the values of different types of variables and to easily control execution, if needed.

Debug Toolbar

Processes can fail to complete successfully if erroneous parameter settings or outside events cause operations to fail during processing of a Process. To find run-time errors in Processes, use the Debug toolbar, which is active only when you are editing an instance of a Process. The Debug toolbar is inactive when you are editing a Process object.

Breakpoints

One way to identify errors is to use breakpoints to check values of variables and operation parameters. A breakpoint can be set on an operation to interrupt a Process when processing before the operation is processed. When a breakpoint is set, the execution of the entire Process is suspended when processing reaches the operation with the breakpoint. Breakpoints can be set in either the object definition of a Process or in a suspended instance of a Process. When the instance of the Process is created at the beginning of the execution, any breakpoints in the Process object definition are copied to the instance.

Process State

You can also use the UserInstName feature to indicate the state of a CA IT PAM Process at each step of that Process. For example, while processing an event from an Event Management System if the event ID or other information is missing the UserInstName could be set to "Create_Info_Missing". If the information is available, then the UserInstName could be set to "Received_Reqd_Info." This enables you to monitor the state of several running instances within a process.

IT Log Messages

A Log Viewer object monitors log messages generated by Touchpoints and their modules. You can add filters to view actions executed by a specified server, on a specified Agent, and by a module. A Log Viewer object also provides a dynamic or historical view. By default, a filter displays log messages as they occur but you can optionally choose to view log messages occurring within a specified date-time interval.

The CA IT PAM Log Messages can provide clues to the outcome of Execution and reasons for failure. They include hyperlinks to the parameters and results of past invocation of operations and can be helpful for debugging. User log messages can also be used to record phases of a process.   If you are unable to determine the cause of the error by looking at the user logs you can consult the Orchestrator and Agent Server logs.

JMX Console

One way to monitor the health of a CA IT PAM installation is through a JMX Console. To access the JMX Console for your CA IT PAM installation, browse to the following location on the CA IT PAM Server:

http://<ITPAM SERVER>:<ITPAM Port>/jmx-console/

And log in with the systems JBoss credentials.

Note: The password for the JMX-Consoles "admin" user is "admin". To secure the environment it is recommended that this password is changed. More information on how to secure the JMX console can be found on http://community.jboss.org/docs/DOC-12190 and https://jira.jboss.org/jira/browse/SECURITY-31.

Under the jboss.system group there is an mbean called "ServerInfo" which is identified by the type=ServerInfo. Here you can find information such as the amount of available free memory, total memory, active thread count and more. You can also use the JMX Console to constantly monitor the health of the system. For more information on using the console, including information on how to automate system monitoring, consult the *JBoss JMX Console Guide*. This guide can be obtained from the following link:

http://www.jboss.org/community/wiki/JMXConsole.pdf