IBM Software Accelerated Value Program - WebSphere Application Server Configuration Comparison Tool

Tool Author: Dennis Riddlemoser

Agenda

- Acknowledgements
- Configuration Comparison Tool (CCT) – A Brief History
- What is CCT?
- How does it Work?
- Reading CCT Reports
- File System Comparison
- Sample Correction Scripts
- Support and Request for Enhancements

Acknowledgements

- **Ricky Marley** - Original presentation author
- **Scott Kurz** - Efforts in open sourcing WCCT
- **Thomas Hikade** – Contributions in new reports and bug fixes.

CCT – A Brief History

- Visual Configuration Explorer
  - Difficult to use
  - CPU Intensive
  - No Further Enhancements/Updates
- Multiple WebSphere Clients Requested Cluster Comparison Utility
- Additional Requirements Identified
  - Cross Cell Comparisons
  - Configuration Creep
  - Resource Comparisons
  - Cross Version (Migration) Comparisons

What is CCT?

- Lightweight means of comparing WAS configurations for
  - Global Security
    - Trust Association both global and domain
  - Server configurations
    - Application Servers
    - Node Agents
  - Application configurations
  - Resource configurations
- Lightweight file system comparison facility
- Generation of sample wsadmin scripts

What is CCT – Usage Scenarios?

- Identify inconsistent configuration of servers and resources
- Identify “configuration creep”
- Version to version migration
- File system comparison

How Does It Work?

- CCT report and script generation is a two-part process.
- **ConfigDump.py**
  - Creates a configuration dump of target environment(s)
  - Can be run local or remote to targets
- **ConfigReport.py**
  - Generates reports
    - Report configuration controlled by property file(s) and properties passed on command line
    - Utilizes Perl regular expressions
    - Must be run on native Python (only tested on v2.7)
    - Reports generated in HTML format
  - Generates sample wsadmin correction scripts
    - Optional function
    - Correction scripts based on reports run
    - Provides sample jython code to correct configuration differences

How does it Work? - Contents of CCT tar file

- **ConfigReport.py**: Script for creating reports for Traditional profile WebSphere
- **ConfigDump.py:** wsadmin script for dumping configuration
- **ConfigUtils.py**: Utilities script
- **WAuJ.py**: Utilities script
- **WAuJ_utilities.py**: Utilities script
- **ConfigReport.properties**: Sample properties file for defining reports for Traditional profile WebSphere
- **ConfigReportAttributes.properties**: Defines attributes reported in Traditional profile WebSphere reports. It is not recommended to edit this.

How does it Work? - Contents of CCT tar file

- **CollectFileData.sh**: Collects file data from the specified file system
- **ConfigReportFiles.py**: Generates file system comparison reports
- **WASConfigurationComparisonTool.pdf**: Tool documentation

How does it Work? - Installing CCT

- Download the latest CCT files
  - URL: https://www-01.ibm.com/support/docview.wss?uid=swg22010928
  - Filename: cct-YYYYMMDD.tar.gz
- Place all files in the same directory.
- If file system comparison reports are needed, set read and execute attributes on the following file.
  - CollectFileData.sh
- Minimal install also available.

How does it Work? Installing CCT (Minimal Install)

- Minimum install for dumping a configuration, place the following files in the same directory.
  - ConfigDump.py
  - ConfigUtils.py
  - WAuJ.py
  - WAuJ_utilities.py

- Minimum install for running reports and generating scripts, place the following files in the same directory.
  - ConfigReport.py
  - ConfigUtils.py
  - WAuJ.py
  - WAuJ_utilities.py
  - ConfigReportAttributes.properties

How does it Work? Installing CCT (Minimal Install)

- Minimum install for collecting file system data, place the following file on a directory and set read and execute attributes on it.
  - CollectFileData.sh

- Minimum install for running file system reports, place the following files in the same directory.
  - ConfigReportFiles.py
  - ConfigUtils.py
  - WAuJ.py
  - WAuJ_utilities.py

How Does It Work? – Configuration Dump

- Acquires configuration dump (ConfigDump.py)
- Uses AdminConfig object and wsadmin
- Interacts with DMGR on an ND env, no potential for application server impact
- Generates ${CELL_NAME}.cfg output file by default
- May also generate .pyc and .class files (ignore these)
- Supports local and remote modes via wsadmin
  - Local mode (connType NONE) requires read access to file system containing WAS profile
  - Local mode is preferred due to performance reasons
  - Remote mode requires Administrator, Configurator, Operator or Monitor
- Two optional parameter formats
  - Basic can only specify output file name and cell suffix
  - Advanced can additionally specify configuration filtering
- No required parameters, simple invocation
- Do not make configuration changes to while ConfigDump.py is running.

How Does It Work? – Configuration Dump

- Basic syntax parameters
  - output_file is optional, this overrides the default output file name.
  - cell_suffix is optional
    - Overrides the default cell suffix
    - A cell suffix must be unique in report generation
    - output_file must be specified

Local syntax:

wsadmin.(sh|bat) -conntype NONE -f ConfigDump.py [output_file] [cell_suffix]

Remote syntax:

wsadmin.(sh|bat) -port SOAP_CONNECTOR_ADRESS -host HOSTNAME
-f ConfigDump.py [output_file] [cell_suffix]

Optionally -user USER –password PASSWORD may be specified before -f

How Does It Work? – Configuration Dump

- Advanced syntax
  - "parm=value" format with no spaces before or after "="
  - The following should only be specified once
    - file=output_file specifies the output file name
    - suffix=cell-suffix specifies the cell suffix
  - Any number of the following may be specified. Unless a configuration element matches one of the specifications, it will be excluded from the dump. Not advised for users who are not familiar with the configuration repository structure, reports may fail.
    - node=node_name specifies a node scope
    - server=server_name specifies a server scope
    - cluster=cluster_name specifies a cluster scope
    - application=application_ear file specifies an application scope
  - Only one of the following should be specified. It is a comma delimited list of configuration types. Use these to disable direct collection of the specified. Note, indirect collection may still occur.
    - excludeType=Type1,Type1,Type3,…

How Does It Work? – Configuration Dump

Advanced syntax samples

Local syntax:

wsadmin.(sh|bat) -conntype NONE -f ConfigDump.py [file=output_file] [suffix=cell_suffix] [node=node_name] [cluster=cluster_name] [application=application_ear_file]

Remote syntax:

wsadmin.(sh|bat) -port SOAP_CONNECTOR_ADRESS -host HOSTNAME -f ConfigDump.py file=output_file] [suffix=cell_suffix] [node=node_name] …

Optionally -user USER –password PASSWORD may be specified before -f

How Does It Work? – Configuration Dump Output

```
CellSuffix=190605154642
traceEnabled(cells/CCT-Cell-v9|resources.xml#ConfigProperty_1549239875618)
    confidential=false
    ignore=false
    name=traceEnabled
    supportsDynamicUpdates=false
    descriptions=[]
    type=java.lang.Boolean
    description=traceEnabled
(cells/CCT-Cell-v9|virtualhosts.xml#MimeEntry_1549240402090)
    extensions=sxc
    type=application/vnd.sun.xml.calc
(cells/CCT-Cell-v9|virtualhosts.xml#MimeEntry_507)
    extensions=xop
    type=application/xop+xml
(cells/CCT-Cell-v9|ws-security.xml#CallbackHandler_1084441805540)
    properties=[]
    key=["CN=Bob,O=IBM,C=US"(cells/CCT-Cell-v9|ws-security.xml#Key_1084441805540)]
    keyStore=(cells/CCT-Cell-v9|ws-security.xml#KeyStore_1084441805540)
    classname=com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler
(cells/CCT-Cell-v9|virtualhosts.xml#MimeEntry_1549240402620)
    extensions=mqy
    type=application/vnd.mobius.mqy
```

Known Limitations for ConfigDump.py

- On Windows platforms, ConfigDump.py must be run from the directory it is installed in if a Unix style “sh” command is not in the path. This is due to a Jython limitation. The following error is seen.

ERROR, cannot continue. Failed to load support scripts. Run wsadmin -f ConfigDump.py from the same directory it exists in.

- The following message is seen when ConfigDump.py finishes. This is due to wsadmin expecting a AdminConfig.save() to be called. Since ConfigDump.py makes no changes, no save() is called and this message is expected.

WASX7309W: No "save" was performed before the script "ConfigDump.py" exited; configuration changes will not be saved.

How Does It Work? – Report Generation

- Prerequisites to generating reports
- Overview of report generation properties
- Discussion of Perl regular expressions
- Properties processing considerations

How Does It Work? – Report Generation, Prerequisites and Syntax

- Report generation requires Python 2.7
- Requires one or more .cfg files generated by ConfigDump.py

Syntax:

```
python ConfigReport.py arg1 arg2 arg3 arg4...
```

Where arg is either a properties file name or a property in the form of PropertyName=PropertyValue

How Does It Work? – Report Generation, Properties

- Java like properties define report generation
  - Properties file(s) and/or properties are specified on command line
  - Defines reports, input files and output directory
- General properties
  - **ReportList** (optional) is a comma delimited list of reports to generate.
    - "***All***" (default) runs all user defined reports as well as built in reports.
  - **ConfigurationDumpFiles** (optional)is a comma delimited set of configuration dumps. Defaults to *.cfg
  - **ConfigurationReportOutputPath** (optional) defines where reports will be generated. Defaults to the present working directory.
  - **DisableReportKey** (optional) The report key documenting the match column and cell background colors as well as replacement values is not added.

How Does It Work? – Report Generation, Match Properties

- **ReportOnlyMismatched** (optional) do not include items that match.
  - ***"All"*** will also report items that match because of string replacements.
  - ***"True"*** will only report items that mistmatch even with string replacements.

- **DoNotMatchNames** (optional) set to true (default value) to disable replacing cell, node and server names with .* for matching purposes

- **ReplaceForMatch:ID** (optional) Replaces all instances of the value with .*
  - ID is an arbitrary value that must be unique for all ReplaceForMatch instances
  - Changes value matching from equality to regular expression matching
  - Use with caution as this could have unintended results.

- **ReplaceWith:ID** (optional) value is used instead of .* for ReplaceForMatch:ID
  - ID must match a ReplaceForMatch instance ID.

How Does It Work? – Report Generation, Built In Reports

Reports listed below only need to be specified in ReportList. They do not need additional attributes.

- **Summary –** A special report giving server and resource names
- **Resources -** Compares all like resources by exact name match
- **NodeAgent -** Compares all node agent configurations
- **Applications –** Compares all deployed applications by exact name of archive
- **Clusters -** Finds all clusters (using an exact name match) and creates one server report per cluster comparing all cluster members
- **TrustAssociation –** Compares Global and domain trust association configuration as well as interceptor properties

How Does It Work? – Report Generation, Report Properties

#Example configuration report properties

ReportList=All
ConfigurationDumpFiles=*.cfg (Note GLOB regular expressions used, not Perl)
ConfigurationReportOutputPath=c:\temp

#Example configuration report properties

DoNotMatchNames=true
ReplaceForMatch:1=SOME_VALUE
ReplaceForMatch:2=OTHER_VALUE
ReplaceWith:2=VALUE_OF_CHOICE

How Does It Work? – Report Generation, Report Properties

- Report definitions are specially formatted property names.
- The format is NAME:ATTRIBUTE:ID
  - **NAME** an arbitrary label used to associate all attributes with a given report
  - **ATTRIBUTE** is information defining the report
  - **ID** is an arbitrary unique identifier for attributes that may be defined more than once in a report
- Some attributes common to all reports.
  - **ReportType** defines what is being compared
    - Valid values are a subset of valid WAS configuration types
  - **Title** is the title of the HTML report
  - **FileName** is the name of the HTML report on the file system
  - **ReportOnlyMismatched** (optional) overrides global property value of the same name. Same values as global property.

How Does It Work? – Report Generation, ReportType Values

- **Server -** Application server configuration
- **ApplicationDeployment -** Application deployment configuration
- **DataSource -** Data source configuration
- **J2CResourceAdapter -** J2C resource adapter configuration
- **JDBCProvider -** JDBC driver configuration
- **ObjectCacheInstance -** Object cache configuration
- **ServletCacheInstance -** Servlet cache configuration
- **SIBus -** Service Integration Bus configuration
- **ResourceEnvironmentProvider -** Resource environment provider configuration
- **ResourceEnvEntry -** Resource environment entry configuration
- **SSLConfig -** SSL configuration

How Does It Work? – Report Generation, ReportType Values

- **JMSProvider –** JMS Provider
- **GenericJMSConnectionFactory –** Connection, Topic Connection and Queue Connection Factories
- **GenericJMSDestination –** Topics and Queues
- **J2CActivationSpec -** J2C Activation Specifications.
- **JAASAuthData** – JAAS/J2C Authentication Data
- **URLProvider** – Resources / URL Providers
- **URL** – Resources / URLs
- **J2CConnectionFactory** – J2C Connection Factories
- **J2CAdminObject** – Embedded Mess. Queues & Topics
- **MQConnectionFactory,** – WMQ Connection Factories
- **MQQueueConnectionFactory,** – WMQ Queue Connection Factories

How Does It Work? – Report Generation, ReportType Values

- **MQTopicConnectionFactory,** – WMQ Topic Connection Factories
- **MQQueue, MQTopic** – WMQ Queues & Topics

How Does It Work? – Report Generation, Report Properties

- Server reports have two additional attributes
  - **Clusters** selects all servers in matching clusters for comparison
    - Clusters key format is NAME:Clusters:ID
    - Clusters value format is CellRegex::CellRegex
  - **Servers** selects all matching servers for comparison
    - Servers key format is NAME:Clusters:ID
    - There are two server value formats.
    - CellRegex::ServerRegex
    - CellRegex::NodeRegex::ServerRegex
- "Regex" refers to a Perl regular expression
  - CellRegex matches cell names
  - ClusterRegex matches cluster names
  - ServerRegex matches server names
  - NodeRegex matches node names

How Does It Work? – Report Generation, Report Properties

Sample server reports

The following report will compare all servers in the Cluster1 cluster

```
Cluster1:ReportType=Server
Cluster1:Title=Cluster1 Servers
Cluster1:Clusters:1=.*::Cluster1
Cluster1:Servers:1=
Cluster1:ReportList=All
Cluster1:FileName=Cluster1
```

The following report will compare all servers by the name of DCASrv or DCBSrv

```
DCServers:ReportType=Server
DCServers:Title=DC Servers
DCServers:Clusters:1=
DCServers:Servers:1=.*::DCASrv
DCServers:Servers:2=.*::DCBSrv
DCServers:ReportList=All
DCServers:FileName=DC_Servers
DCServers:ReportOnlyMismatched=All
```

How Does It Work? – Report Generation, Report Properties

- Application reports have one additional attribute
  - **Archive** selects all applications by deployed archive name
    - Archive key format is NAME:Archive:ID
    - Archive value format is a regular expression

Sample Application report

App:ReportType=ApplicationDeployment
App:Title=Enterprise Application
App:Archive:1=app1\.ear
App:Archive:2a=app2\.ear
App:Archive:All=.*
App:FileName=AppReport

How Does It Work? – Report Generation, Report Properties

- Resource reports have two additional attributes
  - **Scope** defines the level of the resource to be matched
    - Scope key format is NAME:Scope:ID
    - Scope valid values are Cell, Cluster, Server or Node
  - **Resources** selects the resources by name for comparison
    - Resources key format is NAME:Resources:ID
    - Resources value format is ResRegex
- *Scope* and *Resources* attributes must be in pairs with matching ID
- ResRegex is a regular expression used to match resource names

How Does It Work? – Report Generation, Report Properties

Sample Resource reports

The following report will compare all cluster level Data Sources

```
DS:ReportType=DataSource
DS:Title=Data Sources
DS:Scope:1=Cluster
DS:Resources:1=.*
DS:FileName=DS
```

The following report will compare all node level resource adapters with “MQ” in their name or cluster level with “JMS” in their name

```
J2C:ReportType=J2CResourceAdapter
J2C:Title=Resource Adapters
J2C:Scope:1=Node
J2C:Resources:1=.*MQ.*
J2C:Scope:2=Cluster
J2C:Resources:2=.*JMS.*
J2C:FileName=J2C
```

How Does It Work? – Report Generation, Regular Expressions

- Perl Regular Expressions used for matching
- Common matching syntax
  - . = match any single character
  - * = match previous character 0 or more times
  - + = match previous character one or more times
  - | = logical or operator (normally used with parenthesis)
- Perl Regular Expression documentation available on Internet
- Matching cell names requires special handling
  - Reason is “CellSuffix” is appended on cell name
    - Allows comparisons of cells with same name
    - Allows comparisons of same cell to detect “configuration creep”
  - CellSuffix is found at the beginning of .cfg file
  - CellSuffix has a hard coded delimiter “-” when displayed

How Does It Work? – Report Generation, Properties processing considerations

- There is a single properties name space
  - No conflict checking for properties
  - Last property read takes precedence if a name collision occurs
- Defining properties prior to June 2018
  - There was a single properties file passed in on the command line.
  - All properties were defined in this file
- Defining properties June 2018 to present
  - ConfigReport.py parameters are processed in order
  - ConfigReportAttributes.properties was introduced
    - This file contains lists of which configuration attributes are reported
    - Prior, these properties were part of the single properties file
    - This file loads implicitly and last
    - The file must be in the same directory as ConfigReport.py
    - This will be compatible with report properties files prior to June 2018 if the configuration attribute list properties were not modified

Reading CCT Configuration Reports

- A key is generated at the top of each report
  - Documents color coding conventions and match column values
  - Lists match substitutions

Report Key	
Cell Contents	**Description**
N/A	Only one configuration item, no matching done.
No	At least one item in the row does not match with the rest.
Yes	All values match.
Yes*	All values match with some literals replaced or ignored.
	Background color indicating value is the most common in the row. (Note, there could be multiple values most common).
	Background color indicating all values match due to some literals being repaced.
	Background color indicating value does not match and is not the most common in the row.

Replacement values for matching	
Value	**Is replaced with**
CCT-DM-Node-V8.5.5	.*
CCT-Node1-V8.5.5	.*
CCT-v8.5.5	.*
Cluster1_srv1	.*
Cluster1_srv2	.*
Cluster1_srv3	.*
Cluster2_srv1	.*
Cluster2_srv2	.*
Cluster2_srv3	.*
Cluster2_srv4	.*
dmgr	.*
nodeagent	.*
server1	.*
7778	.*
7777	.*

Reading CCT Configuration Reports

- There is a title for each sub report
- Column headers are names of selected servers or resources
- Row headers are configuration XML attribute values (same as used in scripting configuration updates).

JVM Configuration				
Attribute	**Config Matches**	**Cluster1_srv1 (CCT-Node1-V8.5.5, CCT-v8.5.5-Initial)**	**Cluster1_srv2 (CCT-Node1-V8.5.5, CCT-v8.5.5-Initial)**	**Cluster1_srv3 (CCT-Node1-V8.5.5, CCT-v8.5.5-Initial)**
debugMode	Yes	false	false	false
verboseModeClass	Yes	false	false	false
verboseModeGarbageCollection	No	false	true	true
disableJIT	Yes	false	false	false
maximumHeapSize	Yes	1024	1024	1024
verboseModeJNI	No	true	false	false
bootClasspath	Yes	[]	[]	[]
hprofArguments	Yes	[]	[]	[]
runHProf	Yes	false	false	false
initialHeapSize	Yes	512	512	512
classpath	Yes	[]	[]	[]
debugArgs	Yes*	-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7778	-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777	-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777
internalClassAccessMode	Yes	ALLOW	ALLOW	ALLOW

Reading CCT Configuration Reports

- Match column
  - Green means all config values match for that attribute
    - "Yes" means literal values match
    - "Yes*" means values match with some replaced strings
  - Red means the given attribute does not match on all servers
    - Depends on attribute as to if this is really an issue
      - » For most attributes, it is an issue
      - » Some attributes may contain server specific information which is expected to be different.
  - Yellow (N/A) means only one server or resource was selected so therefore a comparison is not possible.
- Values occurring the least are highlighted in yellow as the probable values in error
- Values that match due to replacement strings are displayed in blue

initialHeapSize	No	512	0	0
classpath	Yes	[]	[]	[]
debugArgs	Yes*	-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777	-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777	-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7792
internalClassAccessMode	Yes	ALLOW	ALLOW	ALLOW

File System Comparison – Collecting Data

- Must be run on locally mounted file system
  - Collect from the same path (relative or fully qualified) on each system
  - Relative paths recommended as fully qualified paths often change from system to system
- File data gathered
  - Path to file
  - Last modified date/time
  - Checksum
- All commands are run via “nice –n10” to ensure CPU is not seriously impacted
- Tested on Linux, Windows and AIX
  - Windows requires GNU utilities in the path (grep, cut, cksum and stat)
  - AIX requires bash in the path
- Output file name defaults to file-list.YYYYMMDD.HHMMSS.txt
  - OutputFileName optionally specified to override default file name
  - Recommended to include date/time stamp in file name, as seen above

Syntax:

CollectFileData.sh FileSystemPath OutputFileName

File System Comparison – Generating Reports

- File system report generation does not require a properties file
- ReportFileName is a file name prefix for the output file
  - Date and time of report generation appended to name
- Any number of outputs from CollectFileData.sh may be specified
  - Each file specified will be a column in the report

Syntax:

python ConfigReportFiles.py ReportFileName FileData1 FileData2 FileData3…

ReportFileName is the name of output report (fully qualified or relative)

FileList parameters are lists of files generated by CollectFileData.sh

Reading CCT File System Reports

- A key is generated at the top of each report
  - Documents color coding conventions and match column values
  - Same look as configuration reports.

Report Key	
Cell Contents	**Description**
N/A	Only one file found, no matching done.
No	At least one file in the row does not match with the rest.
Yes	All files match.
Yes*	File size and CRC match but the date and/or time stamp do not.
	Background color indicating file is the most common in the row. (Note, there could be multiple differnt files equally most common).
	Background color indicating file size and CRC match but the date and/or time stamp do not.
	Background color indicating files do not match and this file is not the most common in the row.

Reading CCT File System Reports

- Column headers are names input files
- Row headers are names of compared files.
- Match column
  - Green means file exists in all file systems
    - "Yes*" means CRC and Size match but date and time differ
  - Red means the file is missing or different (CRC and/or size)

File System Report				
File Name	**Config Matches**	**Data file: FileSystem1.20190604.125417.txt Date: 2019-06-04 Time: 12:54:17**	**Data file: FileSystem2.20190605.105434.txt Date: 2019-06-05 Time: 10:54:34**	**Data file: FileSystem3.txt Date: No Date Time: No Time**
file1.txt	Yes	Date:2019-01-30 Time:18:10:06.384917900 Size:4146723 CRC:3857782564	Date:2019-01-30 Time:18:10:06.384917900 Size:4146723 CRC:3857782564	Date:2019-01-30 Time:18:10:06.384917900 Size:4146723 CRC:3857782564
file2.txt	No	Date:2019-01-31 Time:18:17:36.147083100 Size:4146723 CRC:2231130878	Date:2019-01-31 Time:18:17:36.147083100 Size:4146943 CRC:1132093483	Date:2019-01-31 Time:18:17:36.147083100 Size:4146943 CRC:1132093483
file3.txt	Yes*	Date:2019-01-30 Time:18:23:37.073348600 Size:4146723 CRC:3625702811	Date:2019-01-31 Time:12:46:37.343802600 Size:4146723 CRC:3625702811	Date:2019-01-31 Time:12:46:37.343802600 Size:4146723 CRC:3625702811
file4.txt	No	Date:2019-01-30 Time:18:33:12.747646500 Size:4146722 CRC:2361617172	Date:2019-01-30 Time:18:33:12.747646500 Size:4146722 CRC:2361617172	Date:2019-02-27 Time:14:01:27.774646500 Size:4146722 CRC:277408067
file5.txt	No	NOT FOUND	Date:2019-02-17 Time:05:31:12.465747600 Size:4146722 CRC:2361617172	Date:2019-02-17 Time:05:31:12.465747600 Size:4146722 CRC:2361617172

Sample Correction Scripts

- **OutputScripts** (optional property)
  - ***"True"*** value will generate a sample wsadmin Jython script per report
  - Sample script will synchronize values based on comparison report
  - May be enabled and disabled per report
- Sample script features
  - Commented with "REVIEW" tags to assist in identifying key areas
    - REVIEW tags must be carefully inspected
  - Creates a Repository Checkpoint
  - Creates a section per table with differences in reports
    - Sections labeled with table name
    - Differences are "No" values from the Match Column
  - Defaults to setting most common value from cells in a given row
- Known limitations for generated scripts
  - Variables correction samples not provided due to scope and inheritance issues
  - Samples of creating and deleting properties not provided
  - Using "wsadmin –conntype NONE" highly discouraged

Sample Correction Scripts

Scripts begin with a repository checkpoint

```
# Create a repository checkpoint to revert changes if necessary
print 'Creating repository checkpoint Cluster1_181127093142.py'
AdminTask.createFullCheckpoint('[-checkpointName Cluster1_181127093142.py -checkpointDesc
```

Scripts end with AdminConfig.save() commented out

```
# REVIEW:  Uncomment when all other modifications have been made
#print "Saving configuration changes"
#AdminConfig.save()    # Saves all changes to the master repository
#################################################################
#################### End sample script. ##########################
#################################################################
```

Sample Correction Scripts

Sample Jython to update attributes (sample truncated on right margin)

```
################################################################################
# Sample script for table title "JVM Configuration"
################################################################################
print "Making updates for JVM Configuration"

# REVIEW: Make sure the configuration elements on the servers and nodes listed are correct
configIDs = [ \
# Configuration ID list corresponding to columns in report with "CCT-v8.5.5-Initial" cell name
    '(cells/CCT-v8.5.5/nodes/CCT-Node1-V8.5.5/servers/Cluster1_srv1|server.xml#JavaVirtualMachine_1543097396101)',\
    '(cells/CCT-v8.5.5/nodes/CCT-Node1-V8.5.5/servers/Cluster1_srv2|server.xml#JavaVirtualMachine_1543097411968)',\
    '(cells/CCT-v8.5.5/nodes/CCT-Node1-V8.5.5/servers/Cluster1_srv3|server.xml#JavaVirtualMachine_1543097427296)',\
    ]

# REVIEW: Modify as necessary to select the proper attribute value for the environment
attr_verboseModeGarbageCollection_value = 'true'        #Diffrent values for verboseModeGarbageCollection: "false" "
attr_verboseModeJNI_value = 'false'     #Diffrent values for verboseModeJNI: "false" "true"

for configID in configIDs:
    AdminConfig.modify(configID,'[' +\
        '[verboseModeGarbageCollection "' + attr_verboseModeGarbageCollection_value + '"]' +\
        '[verboseModeJNI "' + attr_verboseModeJNI_value + '"]' +\
        ']')
```

Requests for Enhancements

- CCT is “as is.”
  - Please do not open a case for any issues.
  - Best effort support.
- Submit new enhancement requests or report issues on GitHub: https://github.com/IBM/websphere-cct/issues

Copyright and License