



Installation Guide **QAD Customer Version Control**

70-3393-2.7
QAD Customer Version Control 2.7
April 2019

This document contains proprietary information that is protected by copyright and other intellectual property laws. No part of this document may be reproduced, translated, or modified without the prior written consent of QAD Inc. The information contained in this document is subject to change without notice.

QAD Inc. provides this material as is and makes no warranty of any kind, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. QAD Inc. shall not be liable for errors contained herein or for incidental or consequential damages (including lost profits) in connection with the furnishing, performance, or use of this material whether based on warranty, contract, or other legal theory.

This document contains trademarks owned by QAD Inc. and other companies.

Copyright ©2019 by QAD Inc.

CVC_IG_v027.pdf/sl8

QAD Inc.

100 Innovation Place
Santa Barbara, California 93108
Phone (805) 566-6000
<https://www.qad.com>

Contents

QAD CVC Installation Guide	
Change Summary	v
Chapter 1 Overview	1
QAD CVC Overview	2
System Requirements	2
Chapter 2 Installing, Setting Up, and Upgrading CVC	3
Preparation for Installation	4
Installing Java	4
Installing Git	4
Granting sudo Permission	5
Installing CVC	5
Setting Up CVC	8
Updating the .gitignore File	8
Updating config.ini Settings in CVC	8
Updating Environment Settings in CVC	8
Adding Roles in CVC	11
Adding Users in CVC	12
Defining and Registering Modules in CVC	12
Performing Regular Customization in CVC	20
Upgrading CVC 2.6.1.0 to CVC 2.7.0.60	21
Checking CVC Version	21
Downloading Target CVC Package	21
Identifying and Applying config.ini Changes	21
Identifying and Applying Schema Changes	22
Installing Target CVC Package	23
Initializing File Mappings	24
Upgrading CVC 2.6.0.1 to CVC 2.6.1.0	25
Checking CVC Version	25
Downloading Target CVC Package	25
Identifying and Applying config.ini Changes	25
Identifying Schema Changes	25
Installing Target CVC Package	25
Upgrading CVC 2.5.1.0 to CVC 2.6.0.1	27

Checking CVC Version	27
Downloading Target CVC Package	27
Identifying and Applying config.ini Changes	27
Identifying Schema Changes	28
Installing Target CVC Package	28

Product Information Resources31

QAD CVC Installation Guide Change Summary

The following table summarizes significant differences between this document and previous versions.

Date/Version	Description	Reference
April 2019/2.7.0.60	Updated for 2.7.0.60.	--
	Added instructions for upgrading CVC from 2.6.1.0 to 2.7.0.60.	page 21
December 2018/2.6.1.0	Initial version	--

Overview

This chapter includes an introduction to Customer Version Control (CVC) installation and the prerequisites for installation.

QAD CVC Overview 2

Provides an introduction to QAD CVC.

System Requirements 2

Describes the required tools and applications for installing QAD CVC.

QAD CVC Overview

QAD Customer Version Control, also known as QAD CVC, is a version control system (VCS) for software delivery. It provides a controlled, traceable, and secure process for the maintenance, update, and delivery of software in customer environments.

System Requirements

The following table lists the system requirements for running QAD CVC.

Type	Requirement	Version	Reason
Operating System (OS)	UNIX	N/A	Currently Windows OS is not supported.
Software	Java	1.7 or above	CVC commands were written and compiled using Java. In this release, Java 1.7 or later is required.
Software	Git	1.9.0 or above	CVC uses Git to perform version control on files. Some required <code>git</code> commands are available only in version 1.9.0 or later.
Software	Progress	N/A	CVC uses the Progress database.
CPU		N/A	CVC itself does not require too many resources. Generally, if the system meets Progress and QAD Enterprise Applications requirements, there are sufficient resources to run CVC.
Memory		N/A	
Disk	7 MB + 100 MB + 2 GB	N/A	<p>The size of CVC source code is about 7 MB. The size of a typical database file ranges from 6 MB to 100 MB depending on how many files are version controlled.</p> <p>The size of the <code>.git</code> folder is about 2 GB when CVC operates in a Standard Edition (SE) or QAD Deployment Toolkit (QDT) environment, but about 10 MB when in a YAB environment.</p> <p>Note You do not need to perform version control on standard programs when CVC functions in a YAB environment.</p>

Installing, Setting Up, and Upgrading CVC

This chapter describes the installation, setup, and upgrade of QAD Customer Version Control (CVC).

***Preparation for Installation* 4**

Describes the preparations required for CVC installation.

***Installing CVC* 5**

Describes how to install CVC.

***Setting Up CVC* 8**

Describes how to set up CVC.

***Upgrading CVC 2.6.1.0 to CVC 2.7.0.60* 21**

Describes how to upgrade CVC from 2.6.1.0 to 2.7.0.60.

***Upgrading CVC 2.6.0.1 to CVC 2.6.1.0* 25**

Describes how to upgrade CVC from 2.6.0.1 to 2.6.1.0.

***Upgrading CVC 2.5.1.0 to CVC 2.6.0.1* 27**

Describes how to upgrade CVC from 2.5.1.0 to 2.6.0.1.

Preparation for Installation

Before installing CVC, you must finish the following pre-installation tasks:

- 1 Installing Java
- 2 Installing Git
- 3 Granting sudo Permission

This section lists all the commands you may run in Sudo. You can skip some tasks if the requirement is already met.

The commands you may need to run in the following procedures include:

- `unzip`: To install CVC, you need to run the `unzip` command to extract files to a certain directory, for example, `/usr/local/cvc`, `/opt/cvc`, or `/qond/apps/cvc`.
- `chmod`: To make CVC scripts executable, you need to run the `chmod` command.
- `ln`: To install CVC in the system path, you can create a link by using the `ln` command.

Installing Java

Before installing CVC, make sure that the required Java version (Java 1.7 or later) is installed. If the current Java version is earlier than 1.7, download the required version and install it.

To install Java 1.7 without affecting the Java that has been installed in the system, you can download and extract the `jre` package inside the `cvc tools` folder.

To make CVC use an existing Java version, you can create a link in the `cvc tools` folder to link to the existing `jdk` or `jre` folder. For example, run either of the following commands:

```
ln -s <jdk_path> /usr/local/cvc/java
```

or

```
mkdir /usr/local/cvc/java && ln -s <jre_path> /usr/local/cvc/java/jre
```

Example

To download the `jre.zip` file (Java 1.7) for Linux x64, perform the following steps:

- 1 Run the following commands:

```
$ wget http://packages.qad.com/packages/java-linux64/1.7.0.80/resources  
-O /tmp/jre.zip
```

```
$ ls -l /tmp/jre.zip
```

- 2 Verify that the `jre.zip` file has been downloaded successfully.
- 3 Decompress the package to the `cvc tools` folder.

Installing Git

Before installing CVC, make sure that the required Git version (1.9.0 or later) is installed. If you have not installed Git 1.9.0 or later, download and install it.

The base repository provides Git 1.7.4 only. Therefore, to install Git 1.9.0+ by Yum, you should use the Yum repository in QAD.

Example

To install Git for Linux x64, and RHEL 5 and 6, perform the following steps as user `root`:

- 1 Run the following command:

```
$ vim /etc/yum.repos.d/qad.repo
```

- 2 Fill in the content of `/etc/yum.repos.d/qad.repo` as below:

```
[qad]
name=QAD RPM Repository
baseurl=https://cvc.qad.com/qadrpms/rhel/$releasever/RPMS/$basearch/
gpgcheck=0
enabled=1
```

```
[qad-noarch]
name=QAD RPM Repository
baseurl=https://cvc.qad.com/qadrpms/rhel/$releasever/RPMS/noarch/
gpgcheck=0
enabled=1
```

- 3 Install the Git by Yum.

```
$ yum install git -y
```

- 4 Verify that the Git version is 1.9 or later.

```
$ git --version
git version 2.1.0
```

Granting sudo Permission

CVC runs `sudo` commands to perform Linux user permission control. You need to set the `sudoers` file to allow certain groups or users to run `cvc` commands.

- 1 Modify the `/etc/sudoers.custom` file and add the following content to the end of the file:

```
Cmnd_Alias VC_COMMANDS = /usr/local/cvc/cvc_operator
%qad ALL=(mfg) NOPASSWD: VC_COMMANDS
```

Note No password is required for you to perform this operation. After adding the content, users in the QAD group can run CVC as user `mfg`.

- 2 Generate the `/etc/sudoers` file.

```
$ cat /etc/sudoers.common /etc/sudoers.custom > /etc/sudoers
```

Installing CVC

Generally, before installing CVC, you need to download the CVC package and decompress the files to the `/usr/local/cvc` directory. You also need to create a link in the `/usr/bin` directory so that CVC becomes available in each user's system path.

To install CVC, perform the following steps:

- 1 Download the CVC package.

```
$ wget http://packages.qad.com/packages/cvc/<version>/resources -O /tmp/cvc.zip
```

Note You need to replace `<version>` with the CVC package version you want to install; for example, 2.6.0.1.

6 QAD Customer Version Control Installation Guide

2 Decompress files to the /usr/local/cvc directory.

```
$ unzip /tmp/cvc.zip -d /usr/local/cvc
```

If you want to make CVC use an existing Java version, you can skip the following extract command. However, you need to create a link in the `cvc tools` folder to link to the existing `jdk` or `jre` folder; for example, `ln -sf <jdk_path> /usr/local/cvc/java` or `mkdir /usr/local/cvc/java && ln -s <jre_path> /usr/local/cvc/java/jre`.

```
$ mkdir /usr/local/cvc/java && unzip /tmp/jre.zip -d /usr/local/cvc/java/jre
```

```
$ chmod +x /usr/local/cvc/cvc /usr/local/cvc/cvc_operator  
$ ln -sf /usr/local/cvc/cvc /usr/bin/cvc
```

3 Add required configurations.

```
$ cp -f /usr/local/cvc/etc/man-pages/* /usr/local/share/man/man1 && chmod 644  
/usr/local/share/man/man1/cvc*  
$ chown mfg.qad /usr/local/cvc/etc/config.ini  
$ rm -rf /tmp/cvc.zip && rm -rf /tmp/jre.zip
```

4 Run the `cvc install` command as user `mfg` to install and configure the CVC tool correctly.

Note You need to run the `cvc install` command as user `mfg`, because you may modify the `cvc config.ini` file and create a Git repository in the environment directory.

a Switch to user `mfg`.

```
$ su - mfg
```

b Run the `cvc install` command.

```
$ cvc install
```

```
.....  
Please choose your project type :  
-----  
SEQ Type  
-----  
001 core  
002 qdt  
003 se  
004 yab  
-----  
Please input the sequence number:3  
.....  
Setting up DLC.....  
Current DLC is incorrect [/qad/progress/dlc]  
Please enter DLC path:/progress/dlc  
.....  
Checking database.....  
- Checking database connection.....[fail]  
Unable to setup connection to database.  
Please choose a action
```

```

-----
SEQ Action
-----
001 Connect to database
002 Create a new database
003 Start an existing database
-----

Please choose the sequence number to do the action:2
Please enter the database path:/qadapps/cvc/db/cvcd

.....
Setting environment directory.....
Do you want to run the optional task [Setting environment directory]?
  [Y]es/[N]o:y
Current environment directory is invalid [<initial_dir>]
Please enter environment directory:/qadapps
  - Saving configurations.....[ok]
.....[ok]
Initializing git repository.....
Do you want to run the optional task [Initializing git repository]?
  [Y]es/[N]o:y
  - Saving configurations.....[ok]
.....[ok]
Setting up git remote.....
Do you want to run the optional task [Setting up git remote]?
  [Y]es/[N]o:y
Current git remote [file:///opt/git/repos/central.git] is invalid:
Please enter git remote information:
Git remote url :http://cvc.qad.com/git/ee-lab.git
Git proxy user :ee_devel
Git proxy pass :*****
  - Uploading changes to central server.....[ok]
  - Saving configurations.....[ok]
.....[ok]
CVC is installed successfully!

```

Setting Up CVC

After CVC is installed successfully, you need to set up the CVC tool with the correct environment, module, role, and user.

When setting up CVC, you must finish the following tasks:

- 1 Updating the `.gitignore` File
- 2 Updating `config.ini` Settings in CVC
- 3 Updating Environment Settings in CVC
- 4 Adding Roles in CVC
- 5 Adding Users in CVC
- 6 Defining and Registering Modules in CVC
- 7 Performing Regular Customization in CVC

Updating the `.gitignore` File

To update the `.gitignore` file, perform the following steps:

- 1 Update the `.gitignore` file in the root directory of the environment—for example, `/devl/apps/.gitignore`—to identify the files for which version control is not required. You can also add other files or directories that do not need to be version controlled.
- 2 Commit the `.gitignore` file by running the `cvc commit-git` command:

```
$ cvc commit-git
```

Updating `config.ini` Settings in CVC

You need to update the `jiraStatusAllowed` settings in the `/usr/local/cvc/etc/config.ini` file. This field controls the ticket status relationships with CVC commands. CVC users are only allowed to use commit-related commands with a ticket in the specified status. When configuring the `jiraStatusAllowed` settings, note that this field is case-sensitive and should not have unnecessary spaces.

```
# a mapping of the jira ticket status and checkin/checkout/add commands
# the configure should separate by "command1->status,command2->status,command3->other-
status"
# case sensitive and should not have unnecessary space
jiraStatusAllowed = checkin->Work,checkout->Work,add->Work
```

Updating Environment Settings in CVC

The following are some basic command examples in different environments:

- YAB

You need to configure `Dir` and `Deploy Tool Directory` correctly for each environment. `Deploy Tool Directory` should be set to the one that contains the YAB program. You can obtain the YAB path by running the `which yab` command.

```

$ cvc env -s
These are the environments in system:
-----
Environment  Git branch  Dir                                Deploy Tool Directory              Pre Env  Hotfix
-----
devl         devl        /devl/apps                         /devl/apps/qdt                    test     no
prod         prod        <initial_dir>                      /devl/apps/qdt                    test     no
test        test        <initial_dir>                      /devl/apps/qdt                    devl     no
-----

$ cvc env -u -e devl -tdir /usr/local/yab      # update deploy tool directory for devl environment
Successfully updated environment [devl].
The details of environment [devl]:
-----
Environment  Git branch  Dir                                Deploy Tool Directory              Pre Env  Hotfix
-----
devl         devl        /devl/apps                         /usr/local/yab                    no
-----

$ cvc env -u -e test -dir /test/apps -tdir /qad/local/scripts
$ cvc env -u -e prod -dir /prod/apps -tdir /qad/local/scripts
$ cvc env -s
These are the environments in system:
-----
Environment  Git branch  Dir                                Deploy Tool Directory              Pre Env  Hotfix
-----
devl         devl        /devl/apps                         /usr/local/yab                    no
prod         prod        /prod/apps                         /usr/local/yab                    test     no
test        test        /test/apps                         /usr/local/yab                    devl     no
-----

```

- QDT

You need to configure `Dir`, `Deploy Tool Directory`, and `QDT Env` correctly for each environment. `Deploy Tool Directory` should be set to the installation directory of QDT. `QDT env` refers to the directory that contains the correct `pf` file.

For example, if QDT is installed in the `/devl/apps/qdt` directory and the `batchCompile.pf` file for compiling QAD Enterprise Applications is `/devl/apps/qdt/envs/devldb/scripts/batchCompile.pf`, `QDT env` should be set to `devldb` and the `compile pf` field for the QAD Enterprise Applications module should be `scripts/batchCompile.pf`.

```

$ cvc env -s
These are the environments in system:
-----
Environment  Git branch  Dir                      Deploy Tool Directory      QOT Env  Pre Env  Hotfix
-----
devl         devl        /devl/apps               /devl/apps/qdt            devl     test     no
prod         prod        <initial_dir>           /devl/apps/qdt            prod     test     no
test        test        <initial_dir>           /devl/apps/qdt            test     devl     no
-----

$ cvc env -u -e devl -tdir /devl/apps/qdt --qdt-env devldb # update deploy tool directory for devl environment
Successfully updated environment [devl].
The details of environment [devl]:
-----
Environment  Git branch  Dir                      Deploy Tool Directory      QOT Env  Pre Env  Hotfix
-----
devl         devl        /devl/apps               /devl/apps/qdt            devldb   test     no
-----

$ cvc env -u -e test -dir /test/apps -tdir /test/apps/qdt --qdt-env testdb
$ cvc env -u -e prod -dir /prod/apps -tdir /prod/apps/qdt --qdt-env proddb
$ cvc env -s
These are the environments in system:
-----
Environment  Git branch  Dir                      Deploy Tool Directory      QOT Env  Pre Env  Hotfix
-----
devl         devl        /devl/apps               /devl/apps/qdt            devldb   test     no
prod         prod        /prod/apps               /prod/apps/qdt            proddb   test     no
test        test        /test/apps               /test/apps/qdt            testdb   devl     no
-----

```

- SE

You need to configure `Dir` and `Deploy Tool Directory` correctly for each environment. `Deploy Tool Directory` should be the installation directory of MFGUTIL, where the `langxref.ini` file is located.

```
$ cvc env -s
These are the environments in system:
-----
Environment  Git branch  Dir                                Deploy Tool Directory              Pre Env  Hotfix
-----
devl         devl        /qadapps                          /devl/apps/qdt                    test     no
prod         prod        <initial_dir>                     /devl/apps/qdt                    devl     no
test        test        <initial_dir>                     /devl/apps/qdt                    devl     no
-----

$ cvc env -u -e devl -tdir /qadapps/qea      # update deploy tool directory for devl environment
Successfully updated environment [devl].
The details of environment [devl]:
-----
Environment  Git branch  Dir                                Deploy Tool Directory              Pre Env  Hotfix
-----
devl         devl        /qadapps                          /qadapps/qea                      no
-----

$ cvc env -u -e test -dir /qadapps -tdir /qadapps/qea
$ cvc env -u -e prod -dir /qadapps -tdir /qadapps/qea
$ cvc env -s
These are the environments in system:
-----
Environment  Git branch  Dir                                Deploy Tool Directory              Pre Env  Hotfix
-----
devl         devl        /qadapps                          /qadapps/qea                      no
prod         prod        /qadapps                          /qadapps/qea                      test     no
test        test        /qadapps                          /qadapps/qea                      devl     no
-----
```

Adding Roles in CVC

You can run the `cvc role` command to view all roles in the system. You can also add more CVC roles by using the `cvc role --add` command.

```
$ cvc role -s
These are the cvc roles in system:
-----
SEQ Role      Command
-----
001 admin     encrypt-password,environment,role,user
002 developer  add,check-env,checkin,checkout,config,dev-compile,module-show,report,status
003 releaseManager  add,analyze-rcode,backout,check-consistency,check-env,checkin,checkout,clean-env,config,
dev-compile,module,module-show,promote,register,report,source,status
-----

$ cvc role --add --role reporter --command module-show,report,status
Successfully added role [reporter].
The details of role [reporter]:
-----
SEQ Role      Command
-----
001 reporter  module-show,report,status
-----
```

Adding Users in CVC

You can run the `cvc user` command to view all users who are able to use the `cvc` command. You can also add more CVC users by using the `cvc user --add` command.

```
$ cvc user -s
The details of user [root]:
-----
SEQ Login ID          Full Name          Email              Active Role
-----
001 root              Initial User       who@localhost      Yes  releaseManager,developer,admin
-----

$ cvc user --add --login-id test --full-name "Test User" --email "test@qad.com" --roles "developer"
Successfully added user [test].
The details of user [test]:
-----
SEQ Login ID          Full Name          Email              Active Role
-----
001 test              Test User          test@qad.com       Yes  developer
-----
```

Defining and Registering Modules in CVC

You also need to define and register modules in CVC.

Note Commands involved in the section requires the “release manager” role in CVC. You can run the `cvc user --show` command to see the list of CVC users and user roles.

Defining Modules

Before defining a module in CVC, you need to add the module in CVC and then view module settings in CVC. However, the fields you need to set vary according to the environment type (YAB, QDT, or SE).

- YAB

1 Add the module in CVC.

- Create the directories for the module to be added.

If you want to add a module, you have to create the directories for the module first. For example, before adding module `CUSTOM-MFG`, you need to specify the root directory, work directory, YAB compile process, YAB code instance name, and so on.

Below is an example of the command:

```
$ cvc module --add --module CUSTOM-MFG --root-dir customizations/mfg/default/src --
work-dir customizations/mfg/work/src --yab-process "code-mfg-customizations-update" --
code-instance "mfg-customizations" --before-compile "" --after-compile ""
```

The `yab-process` and `code-instance` fields are dedicated for compiling the code by YAB.

- When compiling customization codes for the YAB project, use `yab-process` of the program files' module. For example, if the module's `yab-process` is set to `code-mfg-update`, run the `yab code-mfg-update` command to compile the program files of the module and update the YAB environment.
- The `code-instance` field is used to generate the compile propath. For example, if the module's `code-instance` is set to `mfg`, you can obtain the module propath by running the `yab config code.mfg.propath` command.
- To have the module compiled correctly by YAB, you must set the `yab-process` and `code-instance` fields correctly.

b Run the `cvc env --show` command to obtain environment information.

The following is an example:

```
$ cvc env --show
These are the environments in system:
-----
Environment  Git branch  Dir                                Deploy Tool Directory          Pre Env  Hotfix
-----
devl         devl       /devl/apps                        /qad/local/scripts            sup      no
prod         prod       /prod/apps                        /qad/local/scripts            test     yes
sup          sup        /sup/apps                         /qad/local/scripts            devl     no
test         test       /test/apps                        /qad/local/scripts            devl     no
-----
```

In the example above, the absolute path of those directories are like the following:

```
root directory: /devl/apps/customizations/mfg/default/src
```

```
work directory: /devl/apps/customizations/mfg/work/src
```

Note Before adding a module, make sure that you have created all the directories required.

c Make sure that the YAB tool is available in the deploy tool directory so that the YAB process can compile module files.

2 View module settings in CVC.

You can run the `cvc module --show` command to view all modules in system. You can also use this command to obtain all the module settings in the system.

The following is an example:

```
$ cvc module --show
```

```
These are the modules in system:
```

Module	Root dir	Work dir	Yab Compile Process	Code Instance	Before Compile	After Compile
CUSTOM-FIN	customizations/fin		code-fin-customizations-update	fin		
CUSTOM-MFG	customizations/mfg/default/src	customizations/mfg/work/src	code-mfg-customizations-update	mfg-customizations		
PATCH-EAM	patches/eam/default/src/		code-eam-update			
PATCH-EAM-DATA	patches/eam/default/data					
PATCH-MFG	patches/mfg/default/src/		code-mfg-update	mfg		
PATCH-MFG-DATA	patches/mfg/default/data/					
PATCH-QXTEND-ADAPTER	patches/qxtend/qxtadpt/default/src		code-qxi-update			
PATCH-QXTEND-OUTBOUND	patches/qxtend/qxoserver/default/src		code-qxo-update			

- QDT

1 Add the module in CVC.

- a Create the directories for the module to be added.

If you want to add a module, you have to create the directories for the module first. For example, to add a module, you need to specify the root directory, unencrypted source directory, encrypted source directory, rcode directory, work directory, the propath for compile, the compile .pf file, the module type, the languages code to compile module, the rcode layout, and the source layout.

Below is an example of the command:

```
$ cvc module --add --module mfgpro_cust --type custom --root-dir mfgpro/xxsrc --src-dir
src -work mfgpro/xxsrc/work -rcode mfgpro -propath
mfgpro/xxsrc/src,mfgpro/xrc,mfgpro/xrc/proxy,mfgpro/fin/proxy.pl,mfgpro/xrc/validation
-pf scripts/cvc/compile_mfgpro_cust.pf --languages us -rl staggered -sl twoletter -bc
" " -ac " "
```

If you have not specified the compile .pf file for the module, CVC does not compile all the files linked to the module.

The src dir, xrc dir, type, compile propath, rcode dir, compile pf, rcode layout, source layout, and languages fields are dedicated for customizing and compiling the code by QDT.

- The src dir and xrc dir fields are used to specify the directories containing unencrypted source code and encrypted code separately. When using the cvc check-consistency command, you can check if the code in those folders is from the same source code.
- The type field is used to identify whether the files in the module can be checked out and made customization. CVC only allows checkout from custom modules.
- The compile propath, rcode dir, and compile pf fields are used for compiling and deploying rcode files by QDT. The content of the compile .pf file is used when you compile the files in the module.

- Three options are available to the `rcode layout` field: `staggered`, `flat`, and `source`. Option `staggered` indicates that the `rcode` is placed into the staggered directories; for example, `us/so/sosomt.r`. Option `flat` indicates that the `rcode` is placed into the destination directory directly; for example, `sosomt.r`. Option `source` indicates that the `rcode` is stored into the source directory.

Note Generally, set the `rcode layout` field of the QDT modules to `staggered`.

- Two options are available to the `source layout` field: `twoletter` and `flat`. Option `twoletter` indicates that the source code files are listed in the two-letter subdirectories; for example, `us/so/sosomt.p`. Option `flat` indicates that the source code files are in the flat source code directory; for example, `sosomt.p`.

Note Generally, set the `source layout` field of the QDT modules to `twoletter`.

- The `languages` field is used for deploying `rcode` files into correct language directories. This field supports a comma-separated list of languages. For example, when `source layout` is set to `twoletter` and `rcode layout` is set to `staggered`, if the `languages` field is set to `us, jp`, for the `sosomt.p` file in the `src` folder, CVC compiles with two languages and deploys the `rcode` file to `us/so/sosomt.r`.

b Run the `cvc env --show` command to obtain environment information.

The following is an example:

```
$ cvc env --show
```

```
These are the environments in system:
```

Environment	Git branch	Dir	Deploy Tool Directory	QDT Env	Pre Env	Hotfix
dev1	dev1	/dev1/apps	/dev1/apps/qdt	dev1		no
prod	prod	/prod/apps	/dev1/apps/qdt	prod	sup	no
sup	sup	/sup/apps	/dev1/apps/qdt	sup	test	yes
test	test	/test/apps	/dev1/apps/qdt	test	dev1	no

In the example above, the root directory is `mfgpro/xxsrc`, and therefore the absolute paths of those directories are like following (you can obtain the report by running the `cvc report -m mfgpro_cust` command):

```
$ cvc report -m mfgpro_cust
```

```
The details of the module:
```

Key	Value
Module	mfgpro_cust
Type	custom

```

Root dir          /devl/apps/mfgpro/xsrc
Src dir           /devl/apps/mfgpro/xsrc/src
Xrc dir          /devl/apps/mfgpro/xsrc/xrc
Work dir         /devl/apps/mfgpro/xsrc/work
Rcode dir        /devl/apps/mfgpro
Compile propath  /devl/apps/mfgpro/xsrc/src,/devl/apps/mfgpro/xrc,
                /devl/apps/mfgpro/xrc/proxy,/devl/apps/mfgpro/fin/proxy.pl,
                /devl/apps/mfgpro/xrc/validation,/devl/apps/mfgpro/src
Compile pf       scripts/batchCompile.pf
Lang             us
Rcode Layout     staggered
Source Layout    twoletter
Before Compile
After Compile

```

Note Before adding a module, make sure that you have created all the directories required.

- c** Make sure that the compile .pf file is available in the deploy tool directory so that QDT can compile module files. The pf file path consists of: *env deploy tool dir + envs + env qdt env name + module compile pf path*. QAD recommends you create a folder *cvc* under the */devl/apps/qdt/envs/devl/scripts* directory to hold all compile .pf files of the CVC module. Then you can set the *compile pf file* field to *scripts/cvc/ compile_mfg.pf*.

The following is an example of the absolute pf file path:

```
compile pf file path : /devl/apps/qdt/envs/devl/scripts/cvc/compile_mfg.pf
```

The following is an example of the content of the pf file:

```

-db /devl/apps/mfgpro/db/mfgempty -ld qaddb -trig triggers -RO
-db /devl/apps/mfgpro/db/srcempty -ld srcode -trig triggers -RO
-db /devl/apps/mfgpro/db/admemory -ld qadadm -trig triggers -RO
-db /devl/apps/mfgpro/db/hlpempty -ld qadhelp -trig triggers -RO
-inp          32000
-tok          4096
-l           1000
-c           500
-s           32000
-D           50
-TM          5
-TB          2
-B           10
-yy          1920
-d           mdy
-h           7
-cprcodeout  utf-8
-cpinternal  utf-8

```

```
-cpstream      utf-8
```

2 View module settings in CVC.

You can run the `cvc module --show` command to view all modules in system. You can also use this command to obtain all the module settings in the system.

The following is an example:

```
$ cvc module --show
These are the modules in system:
-----
Module   Type   Root dir   Src dir Xrc dir Work dir   Rcode dir Compile propath   Compile pf   Lang Rcode Layout Source Layout Before Compile After Compile
-----
mfgpro_cust custom mfgpro/xxsrc src      mfgpro/xxsrc/work mfgpro mfgpro/xxsrc/src, scripts/batchCompile.pf us staggered twoletter
mfgpro/xrc,
mfgpro/xrc/proxy,
mfgpro/fin/proxy.pl,
mfgpro/xrc/validation,
mfgpro/src
mfgpro_src standard mfgpro src xrc      mfgpro mfgpro/xrc, scripts/batchCompile.pf us staggered twoletter
mfgpro/xrc/proxy,
mfgpro/fin/proxy.pl,
mfgpro/xrc/validation
-----
```

- SE

1 Add the module in CVC.

a Create the directories for the module to be added.

If you want to add a module, you have to create the directories for the module first. For example, to add a module, you need to specify the root directory, unencrypted source directory, encrypted source directory, rcode directory, work directory, the propath for compile, the compile `.pf` file, the module type, the languages code to compile module, the rcode layout, and the source layout.

The following is an example of the command:

```
$ cvc module --add --module mfgpro_cust --type custom --root-dir qea/xxsrc --src-dir src
-work qea/xxsrc/work -rcode qea -propath qea/xxsrc/src,qea/xrc,qea/src -rl staggered -
sl flat --languages us -pf compile_mfg.pf
```

If you have not specified the compile `.pf` file for the module, CVC does not compile all the files linked to the module.

The `src dir`, `xrc dir`, `type`, `compile propath`, `rcode dir`, `compile pf`, `rcode layout`, `source layout`, and `languages` fields are dedicated for customizing and compiling the code by MFGUTIL.

- The `src dir` and `xrc dir` fields are used to specify the directories containing unencrypted source code and encrypted code separately. When using the `cvc check-consistency` command, you can check if the code in those folders is from the same source code.
- The `type` field is used to identify whether the files in the module can be checked out and made customization. CVC only allows checkout from custom modules.
- The `compile propath`, `rcode dir`, and `compile pf` fields are used for compiling and deploying `rcode` files by MFGUTIL. The content of the compile `.pf` file is used when you compile the files in the module.

- Three options are available to the `rcode layout` field: `staggered`, `flat`, and `source`. Option `staggered` indicates that the `rcode` is placed into the staggered directories; for example, `us/so/sosomt.r`. Option `flat` indicates that the `rcode` is placed into the destination directory directly; for example, `sosomt.r`. Option `source` indicates that the `rcode` is stored into the source directory.

Note Generally, set the `rcode layout` field of the SE modules to `staggered`.

- Two options are available to the `source layout` field: `twoletter` and `flat`. Option `twoletter` indicates that the source code files are listed in the two-letter subdirectories; for example, `us/so/sosomt.p`. Option `flat` means the source code files are in the flat source code directory; for example, `sosomt.p`.

Note Generally, set the `source layout` field of the SE modules to `flat`.

- The `languages` field is used for deploying `rcode` files into correct language directories. This field supports a comma-separated list of languages. For example, when `source layout` is set to `flat`, and `rcode layout` set to `staggered`, if the `language` field is set to `us, jp`, for the `sosomt.p` file in the `src` folder, CVC compiles and deploys the two different language `rcode` files to `us/so/sosomt.r` and `jp/so/sosomt.r`, separately.

b Run the `cvc env --show` command to obtain environment information.

The following is an example:

```
$ cvc env --show
These are the environments in system:
-----
Environment  Git branch  Dir                               Deploy Tool Directory           Pre Env  Hotfix
-----
devl         devl       /qadapps                         /qadapps/qea                   no
prod        prod       /qadapps                         /qadapps/qea                   sup      no
sup         sup       /qadapps                         /qadapps/qea                   test     yes
test        test       /qadapps                         /qadapps/qea                   devl    no
-----
```

In this example, the root directory is `qea/xxsrc`, and therefore the absolute paths of those directories are like following (you can obtain the report by running the `cvc report -m mfg_cust` command):

```
$ cvc report -m mfg_cust
The details of the module:
-----
Key          Value
-----
Module       mfg_cust
Type         custom
Root Dir     /qadapps/qea/xxsrc
Src dir      /qadapps/qea/xxsrc/src
Xrc dir
```

```

Work dir          /qadapps/qea/xxsrc/work
Rcode dir         /qadapps/qea
Compile propath  /qadapps/qea/xxsrc/src,/qadapps/qea/xrc,/qadapps/qea/src
Compile pf       compile_mfg.pf
Lang             us
Rcode Layout     staggered
Source Layout    flat
Before Compile
After Compile

```

Note Make sure that you have created all required directories before adding a module.

- c** Make sure that the compile .pf file is available in the deploy tool directory so that MFGUTIL can compile module files.

```
compile pf file path : /qadapps/qea/compile_mfg.pf
```

The following is an example of the content of the compile .pf file:

```

-db /qadapps/qea/db/mfgempty -ld qaddb -RO -trig triggers
-db /qadapps/qea/db/admempty -ld qadadm -RO -trig triggers
-db /qadapps/qea/db/hlpempty -ld qadhlp -RO -trig triggers
-db /qadapps/qea/db/audempty -ld qadaud -RO -trig triggers

```

2 View module settings in CVC.

You can run the `cvc module --show` command to view all modules in the system. You can also use this command to obtain all the module settings in the system.

The following is an example:

```

$ cvc module --show
These are the modules in system:
.....
Module   Type   Root dir  Src dir  Xrc dir  Work dir   Rcode dir  Compile propath   Compile pf   Lang  Rcode Layout  Source Layout  Before Compile  After Compile
.....
mfgpro_cust  custom  qea/xxsrc  src      qea/xxsrc/work  qea      qea/xxsrc/src,qea/xrc,  compile_mfg.pf  us    staggered  flat
qea/src
mfgpro_src  standard  qea      src      xrc      qea      qea/xrc,qea/src      compile_mfg.pf  us    staggered  flat
.....

```

Registering Modules

To register a module, perform the following steps:

- 1** Make sure that you have cleaned up the environment. This means no backup files or garbage files are available.
- 2** Run the `cvc register` command to specify the relationships between files and modules. This means you need to identify which files belong to each module.

Note QAD highly recommends that you run the `cvc register` command in simulate mode first.

The following is a sample simulate command that you can run to register all the files in the root directory of the Custom-QAD module:

```
cvc register --ticket CVC-100 --module "Custom-MFG" --note "Simulating register" --
```

```
simulate 'src.*'
```

You can find the simulated whole list once the command is completed.

After verifying the results, you can register the files by removing the `--simulate` option:

```
cvc register --ticket CVC-100 --module "Custom-MFG" --note "Register Custom-QAD" 'src.*'
```

Note When running the `cvc register` command, you register only file information in the CVC database. If you have set incorrect values when running this command, you could run it again with the `--list-file` option:

```
cvc register --ticket CVC-100 --module "Custom-MFG" --note "Correct last register" --list-file register.lst --simulate
```

Example

The following is an example of the `register.lst` file, which is used to register all files under the `root_dir` module:

```
include .*
```

The following is another example of the `register.lst` file, which is used to register only the files in the `src/dc` and `src/pp` directory under the `root_dir` module:

```
exclude .*
include src/dc/*. *
include src/pp/*. *
```

Performing Regular Customization in CVC

This section describes some regular customization operations you can perform when using CVC.

Such operations include but are not limited to:

- Adding customization files

You can run the `cvc add` command to add files in the ticket folder to a specified module.
- Checking out existing files from a customization folder

You can run the `cvc co` or `cvc checkout` command to check out the file that you want to customize and specify where the custom module is located. After checkout, you can find the file in the ticket folder in the same directory layout as in the customer environment.

For example, if you check out the `us/xx/xxsosomt.p` file, the layout in the ticket folder is `HDS-XXX/us/xx/xxsosomt.p`.
- Making some changes for customization

You can modify a file after checking it out.
- (Optional) Compiling files locally

Before checking in a customized file to the customer environment, you can use the `cvc dev-compile` command to compile the file in the current ticket folder and view the compile result. This is an optional operation, and you can skip it.
- Checking in files

You can run the `cvc ci` or `cvc checkin` command to check in a customized file to the customer environment. After checkin, the file in the current ticket folder is removed.

Upgrading CVC 2.6.1.0 to CVC 2.7.0.60

This section guides you in upgrading CVC 2.6.1.0 to CVC 2.7.0.60.

The upgrade involves the following tasks:

- 1 Checking CVC Version
- 2 Downloading Target CVC Package
- 3 Identifying and Applying config.ini Changes
- 4 Identifying and Applying Schema Changes
- 5 Installing Target CVC Package
- 6 Initializing File Mappings

Checking CVC Version

To check the CVC version, run the following command:

```
$ cvc --version
```

The system displays the version of the current CVC; for example, 2.6.1.0.

Downloading Target CVC Package

To upgrade CVC from 2.6.1.0 to 2.7.0.60, you need to obtain the CVC package 2.7.0.60, and then decompress both the old and target CVC packages. For example, you can download the packages for CVC 2.7.0.60 and 2.6.1.0 as `new.zip` and `old.zip` separately.

Run the following commands:

```
$ mkdir ~/cvc_upgrade && cd ~/cvc_upgrade
$ unzip old.zip -d old_cvc
$ unzip new.zip -d new_cvc
```

Identifying and Applying config.ini Changes

In this procedure, you need to back up the old `config.ini` file, modify it, and save it as a new one, `new_config.ini`. Then copy the `new_config.ini` file back after replacing the CVC main program files.

Perform the following steps:

- 1 Back up the old `config.ini` file.


```
$ cp /usr/local/cvc/etc/config.ini ~/cvc_upgrade
```
- 2 Compare the `config.ini` files in the existing and target CVC packages:


```
$ cd ~/cvc_upgrade
$ diff old_cvc/etc/config.ini new_cvc/etc/config.ini
23a24,25
> # default report days
> reportDays = 100
105c107,113
< enforceWholeIndexValidation = true
---
> enforceWholeIndexValidation = false
```

```
> # whether to keep xref in the environment during compile
> keepXref = false
> # whether to validate jira attachment [false, warning, error]
> enforceJiraAttachmentValidation = false
> # jira attachment naming convention (use %s as token for ticket)
> jiraAttachmentName = %s_codereview.*
```

- 3 Save the old `config.ini` file as a new one, namely, `new_config.ini`.

```
$ cp config.ini new_config.ini
```

- 4 Open the `new_config.ini` file, set the required fields correctly, and save the changes. Then copy the file back for the changes to take effect.

```
$ vim new_config.ini
$ cp new_config.ini /usr/local/cvc/etc/config.ini && chown mfg.qad
/usr/local/cvc/etc/config.ini
```

As shown in the example above, you need to set four fields correctly in the `new_config.ini` file:

- Change the value of `enforceWholeIndexValidation` from the default one (`true`) to `false`.

Then by default, CVC only displays a warning message when whole-index issues are found in the files to be committed.

- Set `keepXref` to `false`.

Then by default, CVC does not keep the `xref` files generated during the compilation process. (In YAB environments, the `xref` files are reserved if the `keepXref` flag is set from YAB configurations.)

- Set the `reportDays` option correctly.

This option is newly added, allowing you to set the default report data duration.

- If you want to validate JIRA attachments for the code review output files, you need to set `enforceJiraAttachmentValidation` to `true` and set the `jiraAttachmentName` field correctly.

Identifying and Applying Schema Changes

There are some database schema changes required during the upgrade. You need to test the conversion script completely before running it in live environments.

Note that you must back up the CVC database before running the conversion script in case any error happens.

Note Please update the `sqlexp` command options with your own database user, host, port, and database name. You can easily find the settings in the `/usr/local/cvc/etc/config.ini` file.

- 1 Run the following command to generate the `upgrade.sql` file:

```
$ vim upgrade.sql
```

The following shows the content of the `upgrade.sql` file:

```
drop table cvc_program_reference;
drop table cvc_rcode_analyze_hist;
create table cvc_file_mapping (
  id          bigint          not null,
  file_id     bigint          not null,
  type       int             not null,
  source_name varchar(200)    not null,
  source_path varchar(512)    not null,
```

```

        checksum          varchar(40)    not null,
        last_modified     bigint         not null,
        primary key (id),
        foreign key (file_id) references cvc_file(id)
    ) area "CVC_DATA";
create index idx_file_mapping_fileid on cvc_file_mapping (file_id) area "CVC_INDEX";
create index idx_file_mapping_type on cvc_file_mapping (type) area "CVC_INDEX";
create index idx_file_mapping_path on cvc_file_mapping (source_path) area "CVC_INDEX";
create unique index idx_file_mapping_file_type_path on cvc_file_mapping (file_id, type,
source_path) area "CVC_INDEX";
create unique index idx_file_mapping_file_type_name on cvc_file_mapping (file_id, type,
source_name) area "CVC_INDEX";

create table cvc_file_mapping_hist (
    id          bigint         not null,
    file_id     bigint         not null,
    type        int           not null,
    source_name varchar(200)   not null,
    source_path varchar(512)   not null,
    checksum    varchar(40)    not null,
    last_modified bigint       not null,
    primary key (id),
    foreign key (file_id) references cvc_file_hist(id)
) area "CVC_DATA";
create index idx_file_mapping_hist_fileid on cvc_file_mapping_hist (file_id) area
"CVC_INDEX";

```

2 Run the following command to connect to the database as user mfg.

```

$ sqlexp -user "mfg" -url "jdbc:datadirect:openedge://localhost:30000;databaseName=
cvcdB;defaultSchema=pub;"
Connecting user "mfg" to URL "jdbc:datadirect:openedge://localhost:30000;databaseName=
cvcdB;defaultSchema=pub;"... (8920)

```

3 Run the upgrade.sql file and then exit.

```

SQLExplorer>@RUN upgrade.sql;
SQLExplorer>commit;

SQLExplorer>exit;

```

Installing Target CVC Package

You can update the CVC package with required changes only.

The files that need to be updated are as follows:

- cvc.jar file
- cvc-operator.jar file
- dde-build-core.jar file
- cvc.version file
- Files in the etc/man-pages folder

To install the new CVC package, perform the following steps:

1 Put the previously mentioned files into the system.

```

$ cp -f ~/cvc_upgrade/new_cvc/lib/cvc.jar /usr/local/cvc/lib/cvc.jar
$ cp -f ~/cvc_upgrade/new_cvc/lib/cvc-operator.jar /usr/local/cvc/lib/cvc-operator.jar
$ cp -f ~/cvc_upgrade/new_cvc/lib/dde-build-core.jar /usr/local/cvc/lib/dde-build-
core.jar
$ cp -f ~/cvc_upgrade/new_cvc/cvc.version /usr/local/cvc/cvc.version
$ cp -f ~/cvc_upgrade/new_cvc/etc/schema/* /usr/local/cvc/etc/schema/
$ cp -f ~/cvc_upgrade/new_cvc/etc/man-pages/* /usr/local/cvc/etc/man-pages/
$ cp -f /usr/local/cvc/etc/man-pages/* /usr/local/share/man/man1 && chmod 644
/usr/local/share/man/man1/cvc*

```

2 Verify the CVC version, module settings, and environment settings.

```
$ cvc -v
$ cvc module -s
$ cvc env -s
```

Initializing File Mappings

This release supports a new feature, overlap detecting, as well as some new commands. Therefore you need to run the following commands to initialize the file mappings and update the role permission settings for the new commands.

```
$ cvc role -u -r developer -c "add,backout,check-env,checkin,checkout,config,delete,dev-compile,diff,download,id,module-show,report,resolve,source,status" -e devl
$ cvc role -u -r releaseManager -c "add,analyze-rcode,backout,check-consistency,check-env,checkin,checkout,config,delete,dev-compile,diff,download,file-mapping,id,module,module-show,promote,register,rel-compile,report,resolve,source,status" -e devl
$ cvc role -u -r admin -c "clean-env,commit-git,config,encrypt-password,environment,file-mapping,id,patch,release-lock,revert,role,sql,user" -e devl

$ cvc file-mapping --initialize -e devl
$ cvc analyze-rcode -e devl
$ cvc analyze-rcode -e test
$ cvc analyze-rcode -e prod
```

Upgrading CVC 2.6.0.1 to CVC 2.6.1.0

This section guides you in upgrading CVC 2.6.0.1 to CVC 2.6.1.0.

The upgrade involves the following tasks:

- 1 Checking CVC Version
- 2 Downloading Target CVC Package
- 3 Identifying and Applying config.ini Changes
- 4 Identifying Schema Changes
- 5 Installing Target CVC Package

Checking CVC Version

To check the CVC version, run the following command:

```
$ cvc --version
```

The system displays the version of the current CVC; for example, 2.6.0.1.

Downloading Target CVC Package

To upgrade CVC from 2.6.0.1 to 2.6.1.0, you need to obtain the CVC package 2.6.1.0, and then decompress both the old and target CVC packages. For example, you can download the packages for CVC 2.6.1.0 and 2.6.0.1 as `new.zip` and `old.zip` separately.

Run the following commands:

```
$ mkdir ~/cvc_upgrade && cd ~/cvc_upgrade
$ unzip old.zip -d old_cvc
$ unzip new.zip -d new_cvc
```

Identifying and Applying config.ini Changes

No `config.ini` change is involved for this upgrade. Please skip this procedure.

Identifying Schema Changes

No schema change is involved for this upgrade. Please skip this procedure.

Installing Target CVC Package

You can update the CVC package with required changes only.

The files that need to be updated are as follows:

- `cvc.jar` file
- `cvc-operator.jar` file
- `dde-build-core.jar` file
- `cvc.version` file

- Files in the `etc/man-pages` folder

To install the new CVC package, perform the following steps:

1 Put the previously mentioned files into the system.

```
$ cp -f ~/cvc_upgrade/new_cvc/lib/cvc.jar /usr/local/cvc/lib/cvc.jar
$ cp -f ~/cvc_upgrade/new_cvc/lib/cvc-operator.jar /usr/local/cvc/lib/cvc-operator.jar
$ cp -f ~/cvc_upgrade/new_cvc/lib/dde-build-core.jar /usr/local/cvc/lib/dde-build-core.jar
$ cp -f ~/cvc_upgrade/new_cvc/cvc.version /usr/local/cvc/cvc.version
$ cp -f ~/cvc_upgrade/new_cvc/etc/schema/* /usr/local/cvc/etc/schema/
$ cp -f ~/cvc_upgrade/new_cvc/etc/man-pages/* /usr/local/cvc/etc/man-pages/
$ cp -f /usr/local/cvc/etc/man-pages/* /usr/local/share/man/man1 && chmod 644 /usr/local/share/man/man1/cvc*
```

2 Verify the CVC version, module settings, and environment settings.

```
$ cvc -v
$ cvc module -s
$ cvc env -s
```

Upgrading CVC 2.5.1.0 to CVC 2.6.0.1

This section guides you in upgrading CVC 2.5.1.0 to CVC 2.6.0.1.

The upgrade involves the following tasks:

- 1 Checking CVC Version
- 2 Downloading Target CVC Package
- 3 Identifying and Applying config.ini Changes
- 4 Identifying Schema Changes
- 5 Installing Target CVC Package

Checking CVC Version

To check the CVC version, run the following command:

```
$ cvc --version
```

The system displays the version of the current CVC; for example, 2.5.1.0.

Downloading Target CVC Package

To upgrade CVC from 2.5.1.0 to 2.6.0.1, you need to obtain the CVC package 2.6.0.1, and then decompress both the old and target CVC packages. For example, you can download the packages for CVC 2.6.0.1 and 2.5.1.0 as `new.zip` and `old.zip` separately.

Run the following commands:

```
$ mkdir ~/cvc_upgrade && cd ~/cvc_upgrade
$ unzip old.zip -d old_cvc
$ unzip new.zip -d new_cvc
```

Identifying and Applying config.ini Changes

You need to back up the old `config.ini` file, modify it, and save it as a new one, `new_config.ini`. Then copy the `new_config.ini` file back after applying configuration changes.

Perform the following steps:

- 1 Back up the old `config.ini` file.

```
$ cp /usr/local/cvc/etc/config.ini ~/cvc_upgrade
```

- 2 Compare the `config.ini` files in the existing and target CVC packages:

```
$ cd ~/cvc_upgrade
$ diff old_cvc/etc/config.ini new_cvc/etc/config.ini
91a92,103
> # whether to run xref index analysis tools
> xrefEnabled = true
> # the process-xref script location
> xrefProcessScript =
> # the exemption file location for process-xref script
> xrefExemptionFile =
> # the check-indexes script location
> xrefCheckIndexesScript =
> # the QAD database location for check-indexes script (separated by commas, without the
```

```
.db extension)
> xrefDbLocation =
> # the relative path of the QAD application root directory in YAB, should be inside the
environment directory
> yabEnvRootDir =
> # enforce WHOLE-INDEX validation
> enforceWholeIndexValidation = true
```

- 3 Save the old `config.ini` file as a new one, namely, `new_config.ini`.

```
$ cp config.ini new_config.ini
```

- 4 Open the `new_config.ini` file, set the required fields correctly, and save the changes. Then copy the file back for the changes to take effect.

```
$ vim new_config.ini
$ cp new_config.ini /usr/local/cvc/etc/config.ini && chown mfg.qad
/usr/local/cvc/etc/config.ini
```

As shown in the example above, you need to set seven fields correctly in the `new_config.ini` file:

- Set `xrefEnabled` to `true` to run xref index analysis tools.
- Set `xrefProcessScript`, `xrefExemptionFile`, `xrefCheckIndexesScript`, and `xrefDbLocation` correctly.
- Set `enforceWholeIndexValidation` to `true` to enforce whole-index validation during check-in and registration. If this field is set to `false`, CVC only displays a warning message when whole-index issues are found in the files to be committed.
- If you want to specify the QAD application root directory in YAB as a sub directory of the CVC environment root directory, you need to set `yabEnvRootDir` to the relative path correctly.

Identifying Schema Changes

No schema change is involved for this upgrade. Please skip this procedure.

Installing Target CVC Package

You can update the CVC package with required changes only.

The files that need to be updated are as follows:

- `cvc.jar` file
- `cvc-operator.jar` file
- `dde-build-core.jar` file
- `cvc.version` file
- Files in the `etc/man-pages` folder

To install the new CVC package, perform the following steps:

- 1 Put the previously mentioned files into the system.

```
$ cp -f ~/cvc_upgrade/new_cvc/lib/cvc.jar /usr/local/cvc/lib/cvc.jar
$ cp -f ~/cvc_upgrade/new_cvc/lib/cvc-operator.jar /usr/local/cvc/lib/cvc-operator.jar
$ cp -f ~/cvc_upgrade/new_cvc/lib/dde-build-core.jar /usr/local/cvc/lib/dde-build-
core.jar
$ cp -f ~/cvc_upgrade/new_cvc/cvc.version /usr/local/cvc/cvc.version
```

```
$ cp -f ~/cvc_upgrade/new_cvc/etc/schema/* /usr/local/cvc/etc/schema/
$ cp -f ~/cvc_upgrade/new_cvc/etc/man-pages/* /usr/local/cvc/etc/man-pages/
$ cp -f /usr/local/cvc/etc/man-pages/* /usr/local/share/man/man1 && chmod 644
/usr/local/share/man/man1/cvc*
$ cp ~/cvc_upgrade/new_config.ini /usr/local/cvc/etc/config.ini && chown mfg.qad
/usr/local/cvc/etc/config.ini
```

2 Verify the CVC version, module settings, and environment settings.

```
$ cvc -v
$ cvc module -s
$ cvc env -s
```

3 Update the role permission settings for the two commands introduced in CVC 2.6.0.1, that is, `cvc diff` and `cvc download`.

```
$ cvc role -u -r developer -c "add,backout,check-env,checkin,checkout,config,delete,dev-compile,diff,download,id,module-show,report,source,status" -e devl
```

```
$ cvc role -u -r releaseManager -c "add,analyze-rcode,backout,check-consistency,check-env,checkin,checkout,config,delete,dev-compile,diff,download,id,module,module-show,promote,register,rel-compile,report,source,status" -e devl
```

```
$ cvc role -u -r reporter -c "config,diff,id,module-show,report,status" -e devl
```

```
$ cvc role -u -r developer -c "config,diff,download,id,module-show,report" -e test
```

```
$ cvc role -u -r releaseManager -c "analyze-rcode,backout,check-consistency,check-env,config,diff,download,id,module,module-show,promote,register,rel-compile,report" -e test
```

```
$ cvc role -u -r reporter -c "config,diff,id,module-show,report" -e test
```

```
$ cvc role -u -r developer -c "config,diff,download,id,module-show,report" -e prod
```

```
$ cvc role -u -r releaseManager -c "analyze-rcode,backout,check-consistency,check-env,config,diff,download,id,module,module-show,promote,register,rel-compile,report" -e prod
```

```
$ cvc role -u -r reporter -c "config,diff,id,module-show,report" -e prod
```

4 Run the `cvc role -s` command to verify that the settings are correct.

Product Information Resources

QAD offers a number of online resources to help you get more information about using QAD products.

[QAD Forums \(community.qad.com\)](https://community.qad.com)

Ask questions and share information with other members of the user community, including QAD experts.

[QAD Knowledgebase \(knowledgebase.qad.com\)*](https://knowledgebase.qad.com)

Search for answers, tips, or solutions related to any QAD product or topic.

[QAD Document Library \(documentlibrary.qad.com\)](https://documentlibrary.qad.com)

Get browser-based access to user guides, release notes, training guides, and so on; use powerful search features to find the document you want, then read online, or download and print PDF.

[QAD Learning Center \(learning.qad.com\)*](https://learning.qad.com)

Visit QAD's one-stop destination for all courses and training materials.

*Log-in required

