QAD Enterprise Applications
Enterprise Edition 2016

Configuration and Administration Guide
QAD Enterprise Edition

70-3346-2016EE

QAD Enterprise Applications

Enterprise Edition

March 2016

QAD Inc.
100 Innovation Place
Santa Barbara, California 93108
Phone (805) 566-6000
http://www.qad.com

Table of Contents

Administration Guide

Welcome to the *QAD Enterprise Edition Configuration Administration Guide.*

This guide documents describes the deployment, configuration, and administration of the application and is oriented towards system administrators responsible for the maintenance of the application using the Your Application Builder (YAB) Console, version 1.3.

Getting Started

Getting Started provides step-by-step instructions for installing QAD Enterprise Applications and for accessing the installed application. The QAD Enterprise Applications system administration tool YAB is introduced.

Introduction

Installation creates a new instance of QAD Enterprise Applications.

See *QAD Enterprise Edition Installation Guide* available from http://documentlibrary.qad.com/ for installation instructions.

YAB

YAB is a console application that is used to update and administer a QAD Enterprise Applications instance. YAB requires a Java 7 Runtime Environment (JRE) and will use the first JRE that is located on the PATH or the JRE located by the JAVA_HOME environment variable if set.

> **Show Java Version**
> The following command will display the version of the JRE resolved by the PATH:
>
> ```
> java -version
> ```

After installation the YAB console is available from the root directory of your QAD Enterprise Applications install.

It is recommended that YAB is added to the system PATH environment variable. Having YAB in the path allows you execute YAB commands in the context of an environment from any subdirectory within that environment.

Documentation Conventions / Assumptions

The root directory used in all examples in this document is

`/dr01/qadapps/qea`

It is assumed that 'yab' is in the system PATH.

All paths used in the documentation are relative to the root directory.

Execute YAB to verify your installation.

```
> yab
YAB Console, 1.3.0.32

 Usage: yab <options> [PROCESS ...]

 Options:

 -a:arg          Locates the application to manage.

                 Default: current working directory

 -v              Writes log messages to the console.

 -p:arg          A FILE|URL locating configuration settings to install/update.
                 (multiple allowed)

 -i:arg          A FILE locating a package to install into the local software
                 catalog. (multiple allowed)

 -r              Forces a configuration refresh.

 -clean          Rebuilds the application cache.

 -log-level:arg  Sets a logging threshold [TRACE|DEBUG|INFO|WARN|ERROR|OFF]

                 Default: DEBUG

 Arguments:

 PROCESS         A process to execute.
```

Temp Directory

YAB creates temporary files in the */tmp* directory. Configuration of the temp directory is detailed in the Troubleshooting section of this document.

Clients

The QAD Enterprise Edition environment should be started before accessing a client.

Character

The character client is started by executing a (language specific) script in the installation directory:

Start a character session.

```
> scripts/client-[LANG].sh
```

Ex.

```
> scripts/client-us.sh
```

.NET

The QAD .NET UI client must be installed on a Windows host. The .NET client installation is started by navigating to the Home Server in a web browser.

The YAB Console *env-info* command displays environment information, including the Home Server URL of the current instance:

```
> yab env-info
Client
                                        QAD Home
http://vmlinux.qad.com:22000/qadhome
                                        .NET UI
http://vmlinux.qad.com:22000/qadui
QXtend
                                        Inbound   http://vmlinux.qad.com:22095/qxi
                                        Outbound  http://vmlinux.qad.com:22095/qxo
```

YAB Console

The YAB console is an application that is used to administer QAD Enterprise Applications instances. The YAB console application is installed to the root of a QAD Enterprise Applications instance when that instance is installed.

```
> yab info
```

An understanding of the following fundamental concepts will provide a solid foundation for administering QAD Enterprise Applications with YAB:

- Packages
- Commands
- Configuration
- File System

A single installation of the YAB console may be used to administer multiple QAD Enterprise Applications instances. An instance is selected by using the (-a) option to locate the instance or by navigating into the installation directory of the instance.

```
> yab -a:/dr01/qadapps/devl info
> yab -a:/dr01/qadapps/test info
> cd /dr01/qadapps/devl
> yab info
```

If an instance cannot be located, the following error is raised.

```
> yab info
ERROR - Unable to locate an application.
Use (-a:PATH) option to locate an application.
```

Packages

An environment is composed of *packages*. Packages are read-only software bundles that are used to distribute application and management components.

Software Catalogs

A software catalog is a structured collection of packages. A catalog may be stored in compressed (ZIP) or uncompressed format. Every QAD Enterprise Applications instance is associated with a software catalog that stores packages in uncompressed format.

```
> ll build/catalog
total 8
-rw-rw-r--  1 mfg qad    0 Mar 23 17:23 expanded
drwxrwxr-x 79 mfg qad 8192 Mar 23 17:30 packages/

> ll build/catalog/packages/fin-src-proxy/2016/0/80/5/
total 124
drwxrwxr-x   3 mfg qad   4096 Mar 23 17:28 com/
-rw-rw-r--   1 mfg qad     12 Mar 23 17:28 fin-src-proxy.version
-rw-rw-r--   1 mfg qad    313 Mar 23 17:27 metadata.xml
drwxrwxr-x 469 mfg qad 110592 Mar 23 17:28 proxy/
drwxrwxr-x   3 mfg qad   4096 Mar 23 17:28 us/
```

> Packages should never be directly modified.

The *catalogs setting* setting can used to define additional software catalogs to source packages.

```
catalogs=/dr01/qadapps/catalog,/dr01/3rdparty/catalog
```

Downloading Packages

The QAD Enterprise Applications instance must have read access to all configured packages in uncompressed format. If a package is only available in a catalog accessed using the HTTP protocol or that stores packages in compressed format, the package will be downloaded and installed into the local catalog.

Application Usage

QAD Enterprise Applications directly accesses files in the catalog. For example, the default configuration of Tomcat will start Tomcat using files in the *tomcat* package.

See:

Info

Clean

Product Upgrades

Commands

A QAD Enterprise Applications environment is administered by executing *commands* from the YAB console. Commands can be executed individually or chained together.

Ex. Individual

```
> yab database-qaddb-start
> yab database-qadadm-start
```

Ex. Chained

```
> yab database-qaddb-start database-qadadm-start
```

Some commands are used to group together related commands, for example the command to start all databases.

```
> yab database-start
```

The commands that are available in an environment depend on the packages that are installed in the environment. For example, commands to manage the Customer Relationship Management application server (e.g. *appserver-crm-trim*) are only available when Customer Relationship Management is installed.

See:

help

system-process-list

Configuration

Administrative settings are defined in configuration files in the following directory:

`build/config`

Type	Location	Description
Instance Document	build/config/configuration.properties	Used to make changes to the standard configuration. The following procedure is used to apply updates to the system.
System Documents	build/config/system	Documents that define a standard configuration of the QAD Enterprise Applications. WARNING: These files should not be directly edited.
Package Documents	build/config/packages	Factory default settings defined by packages. WARNING: These files are generated and should not be directly edited.

The only configuration file that should ever be modified is *build/config/configuration.properties*.

The fully resolved settings are written to the following file for reference purposes:

`build/work/system/config/current/application.properties`

See:

config

help

update

validate

Configuration Files

File System

The factory default configuration creates an environment where all resources are nested within the installation directory.

```
> ll
total 44
drwxrwxr-x  6 mfg qad  4096 Mar 17 17:23 build/
drwxrwxr-x 11 mfg qad  4096 Mar 17 19:03 config/
drwxrwxr-x  9 mfg qad  4096 Mar 17 17:32 customizations/
drwxrwxrwx  4 mfg qad 12288 Mar 17 05:56 databases/
drwxrwxr-x 13 mfg qad  4096 Mar 17 08:10 dist/
drwxrwxr-x  7 mfg qad  4096 Mar 17 17:32 patches/
drwxrwxr-x  2 mfg qad  8192 Mar 17 13:31 scripts/
drwxrwxr-x  3 mfg qad  4096 Mar 17 17:32 servers/
```

The default layout can be adjusted when the environment is created by following this procedure, referencing the table below.

Factory Default Path	Configuration Setting	Description
.	appdir	The installation directory that was chosen when the application was installed.
build	appdir.build	Used by the management components.
build/catalog	appdir.catalog	The local software catalog.
build/config	appdir.config	The management configuration settings.
build/logs	appdir.logs	The server log files.
build/work	appdir.work	Work area for the management components.
customizations	customizations.dir	Application customizations.
databases	db._base.dir	The databases.
dist	dist.dir	The compiled application binaries.
patches	patches.dir	Application patches.
scripts	scripts.dir	Shell scripts to manage application components.

System Administration

System Administration covers day-to-day administrative tasks such as starting and stopping environments, backup up and restoring databases, and recompiling code.

Environment Management

The YAB console commands start, stop, and status are used to control servers in the environment.

Scripts

Many of the YAB console control related commands execute scripts in the *scripts* directory where the name of the script matches the name of the command. For example the YAB console command *appserver-fin-start* executes *scripts/appserver-fin-start.*

scripts/appserver-fin-start

```
#!/bin/sh

# Starts the 'fin' application server.
# Generated: 2015-03-24T08:04-0700

DLC=/progress/dlc; export DLC
PROMSGS=/progress/dlc/promsgs; export PROMSGS
PATH=/progress/dlc/bin:${PATH}; export PATH
PROTERMCAP=/progress/dlc/protermcap; export PROTERMCAP

/progress/dlc/bin/asbman -i as-fin -start -port 22001
```

The scripts should not be directly edited. Scripts are re-generated by executing the *script-update* (or *update)* comm and from the YAB console.

> It is better to execute YAB console commands than to directly execute scripts. When the command is executed from the YAB console the request and all output generated from the script is recorded in the YAB log file.

Manual Setting

The *manual* setting of each server configures whether the server will (false) or will not (true) be controlled by the environment start, stop, and status commands:

Server	Setting	Default
Database Server	`dbserver.[INSTANCE].manual`	False
Application Server	`appserver.[INSTANCE].manual`	False
WebSpeed Server	`ws.[INSTANCE].manual`	False
Tomcat Server	`tomcat.[INSTANCE].manual`	False
Financials Daemons	`fin.daemons.manual`	False

A "manual" server can still be controlled using server specific YAB console commands. For example, a custom database with the following configuration:

```
db.custom.physicalname=custom
db.custom.manual=true
...
```

Could be controlled with the following commands:

```
> yab database-custom-start
> yab database-custom-stop
> yab database-custom-status
```

Database Backups

Backing up Databases

To backup all databases, execute the command:

```
> yab database-backup
```

Alternatively to backup a specific database, execute the command:

```
> yab database-[INSTANCE]-backup
```

> A databases may be backed up when it is offline or online.

Configuring

A database backup is associated with a *tag* which is a name to identify the backup. By default backups are associated with the tag "default", but this can be overridden with the request.

```
> yab -tag:grs-install database-backup
```

If the tag is not associated with an existing backup of the database the database backup will be added to the tag, otherwise it will be replaced.

Database after imaging can be enabled after the backup if the database is offline.

```
> yab -enableai database-backup
```

The following settings configure the operation of the backup command.

```
# The directory to store backups in.
# Default: APPLICATION/backups
dbbackup.dir=

# When true backups will be GZIP compressed.
# Note: Compression uses platform specific commands (mkfifo, gzip)
and shell syntax.
#       On Windows this setting is ignored.
# Default: true
dbbackup.compress

# When true backups will be associated with a timestamp derived
tag unless a
# the backup request specifies a tag, otherwise when false
requests that do
# not specify a tag are associated with the tag 'default'.
# Default: false
dbbackup.timestamp=

# The maximum number of timestamped backup tags to allow.
# When a new backup is requested and the maximum number of
timestamped backup
# tags is defined, the oldest timestamped backup tag will be
removed.
dbbackup.timestampmax=
```

Restoring Databases

To restore all databases from the default tag, execute the command:

```
> cd /dr01/qadapps/qea
> ./yab database-restore
```

or from a specific tag:

```
> ./yab -tag:NAME database-restore
```

Alternatively to restore a specific database from the default tag, execute the command:

```
> ./yab database-[INSTANCE]-restore
```

or from a specific tag:

```
> ./yab -tag:NAME database-[INSTANCE]-restore
```

See Backing up Databases for a discussion of backup tags.

> Databases must be restored when the system is offline.

Listing Database Backups

To display all database backups, execute the command:

```
> yab database-backup-list
```

Alternatively to display the database backups for a specific database, execute the command:

```
> yab database-[INSTANCE]-backup-list
```

Ex.

```
> yab database-backup-list

Tag: 20150325072049
-------------------
Location: /dr01/qadapps/qea/build/work/backups/20150325072049

hlpdb                Mar 25, 2015 7:21:37 AM

Tag: default
------------
Location: /dr01/qadapps/qea/build/work/backups/default

hlpdb                Feb 25, 2015 1:38:03 PM
admdb                Feb 25, 2015 1:15:28 PM
```

Marking Databases as Backed up

Progress records the last time a database was backed up and uses this metadata in certain contexts (enabling after imaging) to validate requests. Marking a database as backed up manually records the database as having been backed up. It might be appropriate to execute this command after backing up databases using an approach other than Backing up Databases (which automatically marks the backed up databases as backed up).

To mark all databases as backed up, execute the command:

```
> yab database-backup-mark
```

Alternatively to mark a specific database as backed up, execute the command:

```
> yab database-[INSTANCE]-backup-mark
```

> The database must be offline when this command is executed.

Compilation

Compiling

To recompile all source code, execute the command:

```
> yab code-update
```

Alternatively to recompile a specific code base, execute the command:

```
> yab code-[INSTANCE]-update
```

Ex. Recompile the operational code base

```
> yab code-mfg-update
```

Some code sources are associated with multiple sources.

Ex. Multiple sources contribute to the operational code base

```
> yab config code.mfg.*
code.mfg.databases=db.qadcpl,db.qadadm,db.qadhlp,db.qadrcode
code.mfg.dir=/dr01/qadapps/qea/qadapps/qea/dist/mfg
code.mfg.failonerror=true
code.mfg.languages=us,ch
code.mfg.params=-cprcodeout utf-8 -cpinternal utf-8 -T
/dr01/qadapps/qea/qadapps/qea/build/work/tmp -yy 1950 -s 32768 -mmax 8192 -inp 32000
-rereadnolock -c 30 -D 1000 -Bt 350 -nb 200 -cpcoll ICU-UCA -cpcase basic -h 25 -tok
4096 -tmpbsize 8 -TB 31 -TM 32 -cpinternal utf-8 -cpstream utf-8
code.mfg.propath=/dr01/qadapps/qea/qadapps/qea/customizations/mfg/default/src,/dr01/
qadapps/qea/qadapps/qea/customizations/mfg/cust1/src,/dr01/qadapps/qea/qadapps/qea/p
atches/fin/proxypatch,/dr01/qadapps/qea/qadapps/qea/patches/mfg/default/src,/dr01/qa
dapps/qea/qadapps/qea/build/work/archiving/mfg/progress/xrc,/dr01/qadapps/qea/qadapp
s/qea/build/catalog/packages/mfg-ee2016-encrypted/2016/0/16/209/src,/dr01/qadapps/qe
a/qadapps/qea/build/catalog/packages/mfg-ee2016-encrypted/2016/0/16/209/src/validati
on,/dr01/qadapps/qea/qadapps/qea/build/catalog/packages/base-api/1/1/0/79/src,/dr01/
qadapps/qea/qadapps/qea/build/catalog/packages/mfgcoreplus-api/2/17/0/15/src,/dr01/q
adapps/qea/qadapps/qea/build/catalog/packages/qracore-api/2/17/0/32/src,/dr01/qadapp
s/qea/qadapps/qea/build/catalog/packages/requisition-api/1/1/0/91/src,/dr01/qadapps/
qea/qadapps/qea/build/catalog/packages/fin-src-proxy/2016/0/80/5
code.mfg.removeorphanedrcode=true
code.mfg.sources.archiving.dir=/dr01/qadapps/qea/qadapps/qea/build/work/archiving/mf
g/progress/xrc
code.mfg.sources.customizations-cust1.dir=/dr01/qadapps/qea/qadapps/qea/customizatio
ns/mfg/cust1/src
code.mfg.sources.customizations-default.dir=/dr01/qadapps/qea/qadapps/qea/customizat
ions/mfg/default/src
code.mfg.sources.main.dir=/dr01/qadapps/qea/qadapps/qea/build/catalog/packages/mfg-e
e2016-encrypted/2016/0/16/209/src
code.mfg.sources.main.excludes=**/gpsc02.p,**/urltempl.p,**/utdznnc.p,**/wbqu01.p,**
/wbqu02.p,**/wbqu03.p,**/utsequp1.p,**/lvcntora.p,**/gpbrncha.p
code.mfg.sources.main.extra.includes=*.dat,*.ini,version.mfg,**/*.i
code.mfg.sources.patches-default.dir=/dr01/qadapps/qea/qadapps/qea/patches/mfg/defau
lt/src
code.mfg.threads=1
```

These sources can be individually compiled.

Ex. Compile customization 'cust1'.

```
> yab code-mfg-customizations-cust1-update
```

It is better to compile a complete code base and allow YAB to determine what dependent programs need to be recompiled than to compile individual code base sources. The latter approach may result in rcode with inconsistent includes. The commands to recompile individual code base sources are provided to facilitate efficient source code updates.

Configuration

Each code base is associated with settings that configure the compilation.

Ex. Configuration of the operational code base

```
> yab config code.mfg.*
code.mfg.databases=db.qadcpl,db.qadadm,db.qadhlp,db.qadrcode
code.mfg.dir=/dr01/qadapps/qea/dist/mfg
code.mfg.failonerror=true
code.mfg.languages=us
code.mfg.params=-cprcodeout utf-8 -cpinternal utf-8 -T
/dr01/qadapps/qea/build/work/tmp -yy 1920 -s 32768 -mmax 8192 -inp 32000
-rereadnolock -c 30 -D 1000 -Bt 350 -nb 200 -noshvarfix -cpcoll ICU-UCA -cpcase
basic -h 25 -tok 4096 -tmpbsize 8 -TB 31 -TM 32 -cpinternal utf-8 -cpstream utf-8
code.mfg.propath=/dr01/qadapps/qea/customizations/mfg/default/src,/dr01/qadapps/qea/
patches/mfg/default/src,/dr01/qadapps/qea/build/work/avataxeeconnector/mfg/progress/
xrc,/dr01/qadapps/qea/build/work/label-print/erp/code,/qad/local/sandbox/rfrd-cache/
packages/mfg-patch-ee2015/2015/0/15/0/src,/dr01/qadapps/qea/build/work/netui-module-
data-collections-unencrypted/mfg/progress/xrc,/dr01/qadapps/qea/build/catalog/packag
es/mfg-ee2015/2015/0/15/121/src,/dr01/qadapps/qea/build/catalog/packages/mfg-ee2015/
2015/0/15/121/src/validation,/dr01/qadapps/qea/build/catalog/packages/base-api/0/1/0
/441/src,/dr01/qadapps/qea/build/catalog/packages/mfgcoreplus-api/2/11/0/19/src,/dr0
1/qadapps/qea/build/catalog/packages/qracore-api/2/11/0/29/src,/dr01/qadapps/qea/bui
ld/catalog/packages/requisition-api/0/0/0/71/src,/dr01/qadapps/qea/build/catalog/pac
kages/fin-src-proxy/2015/0/80/11
code.mfg.removeorphanedrcode=true
code.mfg.sources.avataxeeconnector.dir=/dr01/qadapps/qea/build/work/avataxeeconnecto
r/mfg/progress/xrc
code.mfg.sources.customizations.dir=/dr01/qadapps/qea/customizations/mfg/default/src
code.mfg.sources.data-collections-unencrypted.dir=/dr01/qadapps/qea/build/work/netui
-module-data-collections-unencrypted/mfg/progress/xrc
code.mfg.sources.label-print.dir=/dr01/qadapps/qea/build/work/label-print/erp/code
code.mfg.sources.main.dir=/dr01/qadapps/qea/build/catalog/packages/mfg-ee2015/2015/0
/15/121/src
code.mfg.sources.main.excludes=**/gpsc02.p,**/urltempl.p,**/utdznnc.p,**/wbqu01.p,**
/wbqu02.p,**/wbqu03.p,**/utsequp1.p,**/lvcntora.p,**/gpbrncha.p
code.mfg.sources.main.extra.includes=*.dat,*.ini,version.mfg,us/bbi/*.i
code.mfg.sources.mfg-patch.dir=/qad/local/sandbox/rfrd-cache/packages/mfg-patch-ee20
15/2015/0/15/0/src
code.mfg.sources.mfg-patch.excludes=**/gpsc02.p,**/urltempl.p,**/utdznnc.p,**/wbqu01
.p,**/wbqu02.p,**/wbqu03.p,**/utsequp1.p,**/lvcntora.p,**/gpbrncha.p
code.mfg.sources.patches.dir=/dr01/qadapps/qea/patches/mfg/default/src
code.mfg.threads=1
```

Configuration Help

To print the documentation for compilation settings:

```
> yab help code.mfg

SETTINGS

     code.aliases                         A comma separated list of database alias
to define of the form:
                                          [ALIAS]=[LOGICAL],[ALIAS]=[LOGICAL]...

     code.compilelist                     Compile using the compile list where paths
are defined relative to the PROPATH.
                                          NOTE: When a compile list is configured,
only the "extra" files are processed when/if
                                          sources (see below) are configured.

     code.databases                       A comma separated list of the databases to
connect.
```

```
                                        Ex. db.qaddb,db.qadadm

     code.debuglist.excludes            A comma-delimited list of exclude patterns
to select the source files to
                                        generate DEBUG-LIST output for.

     code.debuglist.includes            A comma-delimited list of include patterns
to select the source files to
                                        generate DEBUG-LIST output for.

                                        The debug files are generated in the meta
directory. For example, if the
                                        file "us/ie/iehrrp.p" was selected, the
debug list output for the file
                                        would be generated to:

                                        [META DIRECTORY]/debug/us/ie/iehrrp.p

                                        Files that are selected will always be
recompiled.
                                        Debug list output will only be generated
for source files that are not encrypted.

     code.dir                           The directory where the rcode should be
written.

     code.failonerror                   If true the compile should raise an error
when compilation fails.
                                        Default: true

     code.force                         When true all source files will be
(re)compiled, when false (default) an incremental compile will be performed
                                        where source files will be (re)compiled if
they have changed or any files or schema they reference have changed.
                                        (OOABL source code is handled differently
by the Progress compiler. If asked to recompile a class file it
                                        will also recompile all classes and
interfaces referenced from the class.)

                                        NOTE: Under the following conditions a
full compile (force=true) will be done regardless of the value of
                                        this setting:

                                        * The compile request does not record
metadata (nometa=true).
                                        * There is no record of the previous
compile options ([META]/compile.options).
                                        * The record of the relevant previous
compile options (i.e. languages) does not match the current compile
                                        options.

     code.keepxref                      Default: false

     code.languages                     A comma separated list of language
segments to include in the compiled r-code.
                                        Ex. us,du

     code.metadir                       Locates the directory where compile meta
information should be written.
                                        The metadata directory should be dedicated
to the sources associated with this request.
                                        Reference the help on the 'force' setting
for more information on how the metadata
                                        directory is used.

                                        Default: [RCODE DIR]/.meta

     code.nometa                        When true no meta information will be
recorded.
                                        This setting may be set to true as an
optimization when the sources should never be
                                        incrementally compiled to avoid the
overhead of managing meta information.

     code.params                        The progress startup parameters to define.

     code.propath                       The compile PROPATH.

     code.removeorphanedrcode           Removes rcode files that cannot be
```

```
associated with a source file.
                                        Default: false

     code.sources.NAME.dir              The directory containing the source code
to compile.

     code.sources.NAME.excludes         A comma-delimited list of exclude patterns
to select the files to compile.

     code.sources.NAME.extra.excludes   A comma-delimited list of exclude patterns
to select files to copy
                                        from the source directory to the rcode
directory "as is".

     code.sources.NAME.extra.includes   A comma-delimited list of include patterns
to select files to copy
                                        from the source directory to the rcode
directory "as is".

     code.sources.NAME.includes         A comma-delimited list of include patterns
to select the files to compile.
                                        Default: **/*.p,**/*.w,**/*.t,**/*.cls
```

code.threads	The number of threads to use when compiling the code.
	Default: 1

Full Recompile

If a code base is configured to support incremental compilation, source code files will only be recompiled if they or a resource they depend on have changed.

A compile process for a code base that does not support incremental compilation uses the following configuration:

```
code.force=true
```

A full recompile can be forced with the (-force) option.

Ex. Full recompile of the operational code base

```
> yab -force code-mfg-update
```

An alternative approach which starts by first removing the existing rcode.

Ex. Rebuild of the operational code base

```
> yab code-mfg-rebuild
```

Displaying What Would Be Compiled

Using the (-nocompile) option you can display what would be recompiled without actually recompiling the files.

Ex. What if analysis of the operational code base.

```
> yab -nocompile code-mfg-update


code-mfg-update (4 tasks)
--------------------------------------------------------------------------------
1/4 module-avataxeeconnector-stage OK (0.040 s)
2/4 module-label-print-stage OK (0.018 s)
3/4 module-data-collections-unencrypted-stage OK (0.016 s)
4/4 code-mfg-update

SOURCES TO COMPILE
lbcalllps.p
us/lb/lbadhoc.p
us/lb/lbapcfmt.p
us/lb/lbapsvpp.p
us/lb/lbcontmt.p
us/lb/lbcritmt.p
us/lb/lbcrlmrg.p
...


RCODE TO DELETE:
(none)
```

If certain source code files are always showing as needing to be recompiled it is most likely the case that they are encrypted or depend on an encrypted file. The dependency analysis behind an incremental compile cannot be performed on encrypted sourced code.

Generating XREF output

The following configuration setting controls whether XREF files are generated when a code base is compiled.

`code.INSTANCE.nometa`

Ex.

```
code.mfg.nometa=false
```

Normally *nometa* is undefined or set to false to allow the code base to be incrementally compiled. If all or a major portion of the source code is encrypted, *nometa* may be set to true to avoid the wasted overhead of incremental processing. XREF will only be generated for output for source files that are not encrypted. By default the XREF files are deleted when the compile finishes to conserve on disk space. The files can be retained with the *keepxref* setting.

Ex.

```
code.mfg.keepxref=true
```

The XREF files are generated into the directory where the rcode is written, which is configured by the setting *code.INSTANCE.dir*.

`[RCODE DIRECTORY]/.meta/xref`

Ex. XREF for the operational source us/bm/bmasiq.p

`dist/mfg/.meta/xref/us/bm/bmasiq.p`

Generating DEBUG-LIST output

You can configure DEBUG-LIST output to be generated for all source files or for a subset of the source files, using include/exclude patterns to select the files of interest.

Ex. Configure DEBUG-LIST output for the operational source file 'us/bm/bmasiq.p'.

```
code.mfg.debuglist.includes=us/bm/bmasiq.p
```

Ex. Configure DEBUG-LIST output for all the operational source files in the 'us/bm' directory.

```
code.mfg.debuglist.includes=us/bm/*
```

Ex. Configure DEBUG-LIST output for all the MFG/PRO source files.

```
code.mfg.debuglist.includes=**/*
```

To generate the DEBUG-LIST files the source code is compiled:

```
> yab code-mfg-update
```

The DEBUG-LIST files will be generated into the directory where the rcode is written, which is configured by the setting *code.INSTANCE.dir*.

`[RCODE DIRECTORY]/.meta/debug`

Ex. DEBUG-LIST output for the operational source us/bm/bmasiq.p

`dist/mfg/.meta/debug/us/bm/bmasiq.p`

> DEBUG-LIST output will only be generated for source files that are not encrypted. All selected source files will be recompiled (even if they have not changed.)

Auditing

The topics in this section cover the setup and management of database auditing. Consult the *Auditing* chapter in the *QAD Security and Controls* documentation for more information on the configuration of auditing in QAD Enterprise Applications.

Some of the steps described in the *Auditing* chapter have been pre-configured as noted below:

Section	Notes
Enabling Auditing for the Database	The databases are all pre-configured with extents to store auditing data and indexes. See Enable Auditing to enable auditing on databases.
Configuring Database Options and Audit Permissions	The default setup grants audit permissions to the operating system account that created the instance. In a production environment, the guidance in this section would be followed.
Importing Audit Policy	Audit policy files are loaded as described in the *QAD Security and Controls* document ation, but the location of the audit policy files has changed. See Importing Policy Files f or more details.
Creating Optional Archive Database	The system is pre-configured with an archive database (qadarc).
Setting Archive Database Connection	See Manage Archive Database Server for more information on setting up a database server for the archive database.
Customizing Archive/Load Scripts	The scripts have been replaced by the commands described in Archiving Records.
Disabling Auditing	Auditing is disabled on databases using commands that follow the pattern described in Enable Auditing where "enable" is replaced with "disable".

Enable Auditing

To enable auditing on all databases, execute the command:

```
> yab database-auditing-enable
```

Alternatively to enable auditing on a specific database, execute the command:

```
> yab database-[INSTANCE]-auditing-enable
```

The (-deactivateidx) option will deactivate all non-primary indexes for the auditing tables. Deactivating non-primary indexes is available as an option to improve performance when using database auditing. Deactivated indexes may be activated using PROUTIL IDXBUILD. The non-primary indexes are useful for reporting and should be activated on your audit archive database.

```
> yab -deativateidx database-[INSTANCE]-auditing-enable
```

Importing Policy Files

The *Importing Audit Policy* section in the *QAD Security and Controls* documentation has instructions for loading auditing policy files using the program Audit Policy Import (36.12.13.1). The policy files are available at:

```
config/auditing/policies.xml
config/auditing/qadmainpolicy.xml
```

Archiving Records

Before auditing records can be moved from a source database into the archive database, both the source database and the archive database must have auditing enabled.

To move auditing records from all databases enabled for auditing, execute the command:

```
> yab database-auditing-archive
```

Alternatively to move auditing records out of a specific database, execute the following command, where INSTANCE is the name of the source database:

```
> yab database-[INSTANCE]-auditing-archive
```

The archive command consists of two steps. In the first step audit records are exported into a directory within *build/work/auditing* which removes the records from the source database and in the second step the records are loaded into the archive database. The audit records in the work directory are not deleted when the command completes as a precaution to allow the records to be reloaded if the second step fails. If a failure occurs the (-audit.workdir) option can be specified to locate the work directory containing the audit records to load. When the archive process is successful the work directory can be removed.

The following settings configure the operation of the archive command.

```
# The account to use when archiving audit data or empty to use the
current OS account.
#
# NOTE: If an audit.user is not configured the 'Audit
Administrator', 'Audit Data Archiver',
#       and 'Audit Data Reporter' roles will be granted to the OS
account that was used to
#       create the database so auditing functionality works out of
the box. If an audit.user
#       is configured we assume we are working in a restricted
environment where auditing
#       authorization will be handled directly using the Progress
data dictionary.
audit.user=

# The password for the audit.user account.
audit.password=

# The database where archived audit records will be stored.
audit.archive.database=db.qadarc

# The directory where exported audit records will be stored.
audit.archive.dir=${appdir.work}/auditing
```

> Using the default configuration, the operating system account of the YAB administrator must be setup as a QAD Enterprise Applications user to configure auditing within the application. The *Configuring Database Options and Audit Permissions* section within *QAD Security and Controls* provides instructions for configuring audit users and roles that provides a better segregation of responsibilities.

Manage Archive Database Server

The database server for the archive database is not configured to start and stop when the environment is started and stopped. This default can be changed by adding the following configuration setting to build/config/configuration.properties:

build/config/configuration.properties

```
dbserver.qadarc.manual=false
```

To start the archive database server:

```
> yab database-qadarc-start
```

To stop the archive database server:

```
> yab database-qadarc-stop
```

Controlling Financials Daemons

The Financials component is associated with background processes called *daemons*.

The following commands start, stop, and get the status of the daemons:

```
daemon-start
daemon-stop
daemon-status
```

By default the daemon processes are started and stopped when the environment is started and stopped. To disable the automatic start of the daemon processes define the following setting:

```
fin.daemons.manual=true
```

Each daemon listed by the *fin.daemons.names* setting will also have commands to individually control the daemon:

```
daemon-[DAEMON]-start
daemon-[DAEMON]-stop
daemon-[DAEMON]-status
```

> Unsetting fin.daemon.names will prevent any daemon commands from being defined.

```
> yab help daemon-

PROCESSES

     daemon-balancedaemon-start          Starts the balancedaemon.
     daemon-balancedaemon-status         Checks the status of the balancedaemon.
     daemon-balancedaemon-stop           Stops a Financials daemon.
     daemon-budgetdaemon-start           Starts the budgetdaemon.
     daemon-budgetdaemon-status          Checks the status of the budgetdaemon.
     daemon-budgetdaemon-stop            Stops a Financials daemon.
     ...
```

There are other settings to fine tune the Financials API that is called to control the daemons.

```
> yab config fin.daemons.*
fin.daemons.manual=false
fin.daemons.names=XmlDaemon,BalanceDaemon,BudgetDaemon,CrossCompanyDaemon,EventDaemo
n,HistoryDaemon,ReplicationDaemon,ScanDaemon,TimeoutDaemon,CubeDaemon,ReportDaemon
fin.daemons.onlinehousekeeping=OnlineHousekeeping
fin.daemons.propath=/dr01/qadapps/qea/config,/dr01/qadapps/qea/build/catalog/package
s/qracore/2/17/0/32/qad.qra.core/bin/qracore.pl,/dr01/qadapps/qea/dist/fin/patch,/dr
01/qadapps/qea/dist/fin,/dr01/qadapps/qea/dist/pro/com/mfgpro,/dr01/qadapps/qea/buil
d/catalog/packages/fin-bldata/2016/0/80/5,/dr01/qadapps/qea/build/catalog/packages/f
in-bin64-qadfin/2016/0/80/5/qadfin.pl,.,/dr01/qadapps/qea/build/catalog/packages/qra
-oe11/1/1/166/0/lib/qra.pl,/dr01/qadapps/qea/dist/mfg,/dr01/qadapps/qea/dist/mfg/us,
/dr01/qadapps/qea/dist/mfg/us/bbi,/dr01/qadapps/qea/dist/qxtadpt
fin.daemons.startaction=StartDaemon
fin.daemons.startallaction=StartApplication
fin.daemons.statusaction=DaemonStatus
fin.daemons.statusallaction=DaemonStatus
fin.daemons.stopaction=StopDaemon
fin.daemons.stopallaction=StopApplication

> yab help fin.daemons

SETTINGS

     fin.daemons.manual              Whether Financials daemons will be started and
stopped with the environment (false) or manually (true).

     fin.daemons.names               A comma-delimited list of Financials daemons
for which individual control processes (start, stop, status)
                                     should be provided.
                                     Ex.

XmlDaemon,BalanceDaemon,BudgetDaemon,CrossCompanyDaemon,EventDaemon,HistoryDaemon,Re
plicationDaemon,ScanDaemon,Time
                                     outDaemon,CubeDaemon,ReportDaemon

     fin.daemons.propath             The PROPATH to use when calling the Financials
API to control daemons.

     fin.daemons.startaction         The action to submit when starting a specific
daemon.

     fin.daemons.startallaction      The action to submit when starting all daemons.

     fin.daemons.statusaction        The action to submit when getting the status of
a specific daemon.

     fin.daemons.statusallaction     The action to submit when getting the status of
all daemons (currently this is not used).

     fin.daemons.stopaction          The action to submit when stopping a specific
daemon.

     fin.daemons.stopallaction       The action to submit when stopping all daemons.
```

Report Daemon

The report daemon runs on a Windows host and is not directly controlled by YAB. To make it easier to start and stop the report daemon when the environment starts and stops, YAB defines a set of "placeholder" commands (*daemon-reportdaemon-start, daemon-reportdaemon-stop*) that can be extended to call a script that starts and stops the daemon on the Windows host when the environment is started and stopped.

Ex. Execute an rd-start script when the environment starts and an rd-stop script when the environment is stopped.

```
process.reportwin-start.id=daemon-reportdaemon-start
process.reportwin-start.impl=exec
process.reportwin-start.args=/dr01/qadapps/qea/rd-start
process.reportwin-stop.id=daemon-reportdaemon-stop
process.reportwin-stop.impl=exec
process.reportwin-stop.args=/dr01/qadapps/qea/rd-stop
```

When the report daemon on the Windows host communicates back to the server another daemon is automatically spawned on the UNIX host. The command *daemon-reportdaemon-status* shows the status of this UNIX daemon.

Key Commands

clean

The *clean* command performs cleanup activities.

- Fixes package corruption problems in the local catalog.
- Removes orphaned packages from the local catalog.
- Clears all restrictions that prevent a package from being re-added to the local catalog.
- Reclaims disk space consumed by deleted packages in the local catalog.
- Removes orphaned package configurations.
- Removes orphaned lock and PID files.

code-mfg-update

Compiles the operational source code including customizations and patches.

Displays the configuration of the operational compile:

```
> yab config code.mfg.*
```

config

The *config* command queries configuration settings.

```
> yab config | more
adminserver.adminport=22092
adminserver.conmgr=/dr01/qadapps/qea/build/work/generated/conmgr.properties
adminserver.plugins=/dr01/qadapps/qea/build/work/generated/plugins.properties
adminserver.plugins.jvmargs.property=jvmargs
adminserver.plugins.jvmargs.section=PluginPolicy.Progress.AdminServer
adminserver.plugins.jvmargs.value=-Xmx256m -Djava.awt.headless=true
-Dsun.lang.ClassLoader.allowArra
ySyntax=true -Dserver.start.retryCount=10 -Dserver.start.retryInterval=3
adminserver.port=22001
adminserver.ubroker=/dr01/qadapps/qea/build/work/generated/ubroker.properties
aia._base.controllingnameserver=ns-default
aia._base.host=vmdvr02
aia._base.logfile=/dr01/qadapps/qea/build/logs/.log
aia._base.logginglevel=3
aia.default.allowaiacmds=1
aia.default.context=aia
...
```

Examples

Command	Description
`yab config openedge.dir`	Display the Progress Runtime (DLC).
`yab config environment.id`	Display the environment ID.
`yab config "package.*"`	Display package settings.
`yab config "*port"`	Display the TCP-IP ports used.
`yab config "db.*"`	Display database settings.
`yab config "dbserver.*"`	Display database server settings.
`yab config "adminserver.*"`	Display admin server settings.
`yab config "ns.*"`	Display name server settings.
`yab config "appserver.*"`	Display application server settings.
`yab config "ws.*"`	Display webspeed server settings.
`yab config "wsa.*"`	Display web services adapter settings.
`yab config "aia.*"`	Display application internet adapter settings.
`yab config "tomcat.*"`	Display tomcat server settings.
`yab config "webapp.*"`	Display web application settings.
`yab config "code.*"`	Display code compilation settings.
`yab config "*work*dir"`	Display the work directory settings.

The (-trace) option is used to see how a setting was resolved to its current value.

```
> yab -trace config languages
languages=us,ch
  us,ch : build/config/system/environment.properties
  us :
build/config/packages/yab-ee-foundation/1/3/0/50/important/yab-ee-foundation.propert
ies
```

fin-sync-run

The *fin-sync-run* command runs Financials synchronization.

Ex. Runs a full synchronization

```
> yab fin-sync-run
```

Ex. Runs a specific topic.

```
> yab -topic:Topic15 fin-sync-run
```

help

The *help* command displays documentation for commands and configuration settings.

When executed without any arguments the most important management commands are displayed:

```
> yab help

PROCESSES

     clean           Removes temporary files.
     config          Prints configuration settings.
     help            Prints help for configuration settings and processes.
     info            Prints diagnostic information.
     reconfigure     Updates the environment configuration files.
     start           Starts the environment.
     status          Checks the status of servers.
     stop            Stops the environment.
     update          Updates the environment.
     validate        Validates the environment.
```

If an argument is supplied and it exactly matches the name of a process or a configuration setting, the documentation for that resource will be displayed.

```
> yab help config

PROCESS
     config - Prints configuration settings.

DESCRIPTION
     Called without any arguments, all configuration settings with a value will be
printed  (and the option '-all' will include the configuration
     settings without a value). Otherwise, arguments can be used to select the
settings to print. The meta character '*'  is used to match 0 or more
     characters.

     TIP: When using meta characters, quote the argument to prevent the shell from
processing the meta character.

     Ex.

     packages.qracore.version

     "packages.*.version"

     "packages.*"

OPTIONS

     Name                Description
     -----               -----------
     all                 When true includes settings without a value.

                         Default: false

     skip-resolution     When true references are not resolved.

                         Default: false

     skip-value          When true values will not be printed.

                         Default: false

     trace               When true additional information is printed to show how a
configuration setting was resolved.

                         Default: false
```

```
> yab help appserver.crm.srvrloggінglevel

SETTINGS

     appserver.srvrloggінglevel     Logging level for messages into the server log
file.
                                    0 - No log file written
                                    1 - Error only
                                    2 - Basic
                                    3 - Verbose
                                    4 - Extended
                                    This property can be dynamically updated.
Dynamic changes affect
                                    both current and new brokers and/or agents.
```

Summary help is displayed for all commands and configuration settings that start with the argument.

```
> yab help database-qaddb

PROCESSES

     database-qaddb-ai-archiver-disable     Disables the AI archiver for the 'qaddb'
database.
     database-qaddb-ai-archiver-enable      Enables the AI archiver for the 'qaddb'
database.
     database-qaddb-ai-archiver-start       Starts the AI archiver daemon for the
'qaddb' database.
     database-qaddb-ai-archiver-stop        Stops the AI archiver daemon for the
'qaddb' database.
     database-qaddb-ai-disable              Disables AI on the 'qaddb' database.
     database-qaddb-ai-enable               Enables AI on the 'qaddb' database.
     database-qaddb-ai-list                 Lists AI extents on the 'qaddb'
database.
     database-qaddb-ai-status               Checks whether AI is enabled for the
'qaddb' database.
     database-qaddb-ai-switch               Switches to the next AI extent on the
'qaddb' database.
     database-qaddb-auditing-archive        Moves audit records in the 'qaddb'
database to the archive database.
     database-qaddb-auditing-disable        Disables auditing on the 'qaddb'
database.
     database-qaddb-auditing-enable         Enables auditing on the 'qaddb'
database.
     database-qaddb-backup                  Creates a backup of the 'qaddb'
database.
     database-qaddb-backup-list             Lists the backup tags containing backups
of the qaddb database.
     database-qaddb-backup-mark             Marks 'qaddb' database as backedup.
     database-qaddb-create                  Creates the 'qaddb' database.
     database-qaddb-data-update             Loads data into the 'qaddb' database.
     database-qaddb-index-rebuild           Perform an index rebuild.
     database-qaddb-log-print               Prints the 'qaddb' log.
     database-qaddb-rebuild                 Rebuilds the 'qaddb' database.
     database-qaddb-remove                  Removes the 'qaddb' database.
     database-qaddb-restore                 Restores a backup of the 'qaddb'
database.
     database-qaddb-schema-update           Applies schema to the 'qaddb' database.
     database-qaddb-start                   Starts the 'qaddb' database.
     database-qaddb-status                  Checks the status of the 'qaddb'
database.
     database-qaddb-stop                    Stops the 'qaddb' database.
     database-qaddb-structure-list          Prints the structure of the 'qaddb'
database.
     database-qaddb-truncate                Truncates the before image of the
'qaddb' database.
     database-qaddb-update                  Updates the 'qaddb' database.
     database-qaddb-users                   Lists users connected to the 'qaddb'
database.
```

```
> yab help appserver.mfg

SETTINGS

     appserver.agentdetailtimeout                Specifies the timeout value in
seconds used for the asbman
                                                 -agentdetail command. This timeout
value will prevent the asbman /
                                                 wtbman utility from waiting
forever for the agent to respond when
                                                 agent is busy processing a
request. Minimum value is 3 seconds.

     appserver.aia                               The AIA instance to service AIA
connections.

     appserver.aiaurl                            The connection URL when AIA is
used.

     appserver.allowruntimeupdates               0 - to not allow certain
properties to be dynamically updated
                                                 1 - to allow certain properties to
be dynamically updated

     appserver.appserverkeepalivecapabilities    A comma separated list of
keepalive capabilities the SonicMQ Broker
                                                 Connect Adapter responds with when
a client connects to the Adapter and
                                                 requests the keepalive feature.
Both client and server must specify
                                                 "allow" to enable the feature.
                                                 denyServerASK  - disables server
initiated keepalive
                                                 allowServerASK - enables server
initiated keepalive
                                                 denyClientASK  - currently unused
- for future development
                                                 allowClientASK - currently unused
- for future development
...
```

To display all commands regardless of visibility use the (-all) option.

```
> yab -all help | more

PROCESSES

     adminserver-log-print                                          Prints the
AdminServer log.
     adminserver-plugins-configure                                  Configures
the AdminServerPlugins.property file.
     adminserver-rebuild                                            Rebuilds the
AdminServer.
     adminserver-remove                                             Removes the
AdminServer.
     adminserver-script                                             Generates
scripts to control the AdminServer.
     adminserver-start                                              Starts the
AdminServer.
     adminserver-status                                             Checks the
status of the AdminServer.
     adminserver-stop                                               Stops the
AdminServer.
     adminserver-update                                             Updates the
AdminServer.
     aia-default-log-print                                          Prints a
file.
     aia-default-rebuild                                            Rebuilds the
'default' internet adapter.
     aia-default-remove                                             Removes the
'default' internet adapter.
     aia-default-status                                             Checks the
status of the 'default' internet adapter.
     aia-default-update                                             Updates the
'default' internet adapter.
     aia-log-print                                                  Prints the
log of all internet adapters.
     aia-rebuild                                                    Rebuilds all
internet adapters.
     aia-remove                                                     Removes all
internet adapters.
     aia-status                                                     Checks the
status of all internet adapters.
     aia-update                                                     Updates all
internet adapters.
     aim-client-us-update                                           (Re)generates
a file.
...
```

info

The *info* command displays the packages configured in an environment.

The third column will have the value *local* if the package is referenced from the local software catalog and *remote* if it is referenced from a remote software catalog.

```
> yab info

INSTANCE

/dr01/qadapps/qea

MODULES

archiving                                          1.0.0.78       local
assistance-help-ee                                 2016.0.0.2     local
base-api                                           1.1.0.79       local
dde-ant                                            2.2.0.1        local
dde-build                                          2.2.0.49       local
dde-core                                           2.2.0.14       local
dde-registry                                       2.2.0.4        local
demo-data-ee2016                                   2016.0.16.209  local
fin-bin64-proxy                                    2016.0.80.5    local
...
```

The -more option will include more detailed information.

Ex.

```
> yab -more info
```

netui-pro-update

Compiles the Desktop source code including customizations and patches.

Displays the configuration of the Desktop compile:

```
> yab config netui.pro.*
```

reconfigure

The *reconfigure* command updates application configuration files and database settings.

```
> yab reconfigure
```

restart

The *restart* command is command that executes an environment stop followed by an environment start.

shell

The *shell* command starts an interactive shell in which commands can be entered and executed with TAB auto-completion. The shell is useful for learning commands.

Starting Shell

```
> yab shell
** CTRL-D to exit the shell **
enterprise-edition>
```

The shell prompt (test in the example) is set to the value of the *environment.id* setting:

```
enterprise-edition> config environment.id
environment.id=enterprise-edition
0.705 s
enterprise-edition>
```

Executing Commands

In the shell the TAB key will show possible completions for the text entered and when only one possible completion exists will automatically fill in the remaining characters.

```
enterprise-edition> m
metadata-fincore-update          metadata-mfgcoreplus-update
metadata-qracore-update
metadata-update                  metadata-validate               module-adg-stage
module-adg-update                module-archiving-stage
module-archiving-update
module-cmr-stage                 module-cmr-update               module-ctd-stage
module-ctd-update                module-fin-stage                module-fin-update
module-fincore-update            module-fss-stage                module-fss-update
module-grs-stage                 module-grs-update
module-kanban-stage
module-kanban-update             module-ltwb-stage               module-ltwb-update
module-mfg-stage                 module-mfg-update
module-mfgcoreplus-update
module-mrc-stage                 module-mrc-update
module-mswpsw-stage
module-mswpsw-update             module-periodic-costing-stage
module-periodic-costing-update
module-productstructure-stage    module-productstructure-update
module-qracore-update
module-uca-stage                 module-uca-update               module-uig-stage
module-uig-update                module-wms-stage                module-wms-update
mongodb-backup                   mongodb-backup-list
mongodb-backup-remove
mongodb-rebuild                  mongodb-remove                  mongodb-restore
mongodb-script                   mongodb-start                   mongodb-status
mongodb-stop                     mongodb-update
mongodbreplica-initiate


test> mo
module-adg-stage                 module-adg-update
module-archiving-stage
module-archiving-update          module-cmr-stage                module-cmr-update
module-ctd-stage                 module-ctd-update               module-fin-stage
module-fin-update                module-fincore-update           module-fss-stage
module-fss-update                module-grs-stage                module-grs-update
module-kanban-stage              module-kanban-update            module-ltwb-stage
module-ltwb-update               module-mfg-stage                module-mfg-update
module-mfgcoreplus-update        module-mrc-stage                module-mrc-update
module-mswpsw-stage              module-mswpsw-update
module-periodic-costing-stage
module-periodic-costing-update   module-productstructure-stage
module-productstructure-update
module-qracore-update            module-uca-stage                module-uca-update
module-uig-stage                 module-uig-update               module-wms-stage
module-wms-update                mongodb-backup
mongodb-backup-list
mongodb-backup-remove            mongodb-rebuild                 mongodb-remove
mongodb-restore                  mongodb-script                  mongodb-start
mongodb-status                   mongodb-stop                    mongodb-update
mongodbreplica-initiate          mongodbreplica-update-scripts

enterprise-edition>
```

Command arguments are defined in the same manner as from the OS shell.

```
enterprise-edition> config -trace tomcat.default.httpport
tomcat.default.httpport=22000
  @port +0 :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s
0.886 s
enterprise-edition>
```

The UP/DOWN arrow keys will cycle through the command history.

Exiting Shell

Ctrl+D exits the shell.

start

The *start* command starts the environment.

```
> yab start

                            start (29 tasks)
--------------------------------------------------------------------------------
1/29  adminserver-start                                 STARTED (5.664 s)
2/29  nameserver-start                                  OK (0.000 s)
3/29  database-alerts-start                             STARTED (4.196 s)
4/29  database-bisgen-start                             STARTED (3.980 s)
5/29  database-bisgmenu-start                           STARTED (3.004 s)
6/29  database-configurator-start                       STARTED (4.652 s)
7/29  database-dataaccess-start                         STARTED (2.862 s)
8/29  database-dataexch-start                           STARTED (2.947 s)
9/29  database-qadadm-start                             STARTED (3.117 s)
10/29 database-qadcss-start                             STARTED (3.267 s)
11/29 database-qaddb-start                              STARTED (4.861 s)
12/29 database-qadeam-start                             STARTED (3.435 s)
13/29 database-qadhlp-start                             STARTED (2.578 s)
14/29 database-qxevents-start                           STARTED (2.858 s)
15/29 database-qxodb-start                              STARTED (2.764 s)
16/29 database-tmsdb-start                              STARTED (5.381 s)
17/29 appserver-crm-start                               STARTED (11.859 s)
...
```

The start command will start any servers that are not running and can be run repeatedly without any harm. If a server was not running when the command was processed it will display a STARTED status, otherwise an OK status if it was already running.

> Some of the servers are started in the background. For these servers, STARTED indicates that a request was submitted to start the server. To check on the status of the background process use the status com mand.

Individual servers can be started by executing the appropriate command.

```
> yab appserver-fin-start

                        appserver-fin-start (1 task)
--------------------------------------------------------------------------------
1/1 appserver-fin-start                                 STARTED (11.859 s)
--------------------------------------------------------------------------------
```

status

The *status* command shows the status of environment servers.

```
> yab status

                              status (30 tasks)
------------------------------------------------------------------------------
1/30  adminserver-status                                  STARTED (1.048 s)
2/30  nameserver-status                                   STARTED (0.668 s)
3/30  database-qadadm-status                              STARTED (0.134 s)
4/30  database-qaddb-status                               STARTED (0.032 s)
5/30  database-qadhlp-status                              STARTED (0.018 s)
6/30  database-qxevents-status                            STARTED (0.017 s)
7/30  database-qxodb-status                               STARTED (0.024 s)
8/30  appserver-fin-status                                STARTED (0.444 s)
9/30  appserver-mfg-status                                STARTED (0.428 s)
10/30 appserver-qra-status                                STARTED (0.424 s)
11/30 appserver-qxosi-status                              STARTED (0.423 s)
12/30 appserver-qxoui-status                              STARTED (0.439 s)
13/30 appserver-qxtnative-status                          STARTED (0.471 s)
14/30 webspeed-default-status                             STARTED (0.442 s)
15/30 tomcat-default-status                               STARTED (0.072 s)
16/30 tomcat-qxtend-status                                STARTED (0.008 s)
17/30 aia-default-status                                  STARTED (0.943 s)
...
```

The status command will return STARTED if the server is started and STOPPED if the server is stopped. The status of individual servers can be displayed by executing the appropriate command.

```
> yab appserver-fin-status

                        appserver-fin-status (1 task)
------------------------------------------------------------------------------
1/1 appserver-fin-status                                  STARTED (0.536 s)
------------------------------------------------------------------------------
```

stop

The *stop* command stops an environment.

```
> yab stop

                              stop (19 tasks)
--------------------------------------------------------------------------------
1/19  fin-application-stop                              STOPPED (0.519 s)
2/19  qxtend-stage                                      SKIPPED (0.011 s)
3/19  qxo-services-stop                                 STOPPED (0.196 s)
4/19  tomcat-default-stop                               STOPPED (6.631 s)
5/19  tomcat-qxtend-stop                                STOPPED (6.558 s)
6/19  webspeed-default-stop                             STOPPED (2.420 s)
7/19  appserver-fin-stop                                STOPPED (4.134 s)
8/19  appserver-mfg-stop                                STOPPED (2.290 s)
9/19  appserver-qra-stop                                STOPPED (2.000 s)
10/19 appserver-qxosi-stop                              STOPPED (1.909 s)
11/19 appserver-qxoui-stop                              STOPPED (1.759 s)
12/19 appserver-qxtnative-stop                          STOPPED (1.822 s)
13/19 database-qadadm-stop                              STOPPED (7.110 s)
14/19 database-qaddb-stop                               STOPPED (6.114 s)
15/19 database-qadhlp-stop                              STOPPED (6.075 s)
16/19 database-qxevents-stop                            STOPPED (6.074 s)
17/19 database-qxodb-stop                               STOPPED (6.084 s)
18/19 nameserver-stop                                   OK (0.000 s)
19/19 adminserver-stop                                  STOPPED (11.724 s)
--------------------------------------------------------------------------------
```

The stop command will stop any servers that are running and can be run repeatedly without any harm. If a server was running when the command was processed it will display a STOPPED status, otherwise an OK status if it was not running. Servers are generally stopped in the reverse order of how they are started. Individual servers can be stopped by executing the appropriate command.

```
> yab appserver-fin-stop
                          appserver-fin-stop (1 task)
--------------------------------------------------------------------------------
1/1 appserver-fin-stop                                  OK (0.432 s)
--------------------------------------------------------------------------------
```

system-diagnostics

The *system-diagnostics* command creates an archive that contains the log and configuration files well and the output from executing the *system-info* process with the (-more) option.

The archive is created in the current working directory with the following naming pattern:

```
[APPDIR DIRECTORY NAME]-[TIMESTAMP].zip
```

system-process-list

The *system-process-list* command displays the commands that are associated with a command in execution order.

```
> yab system-process-list update | more

1. qxtend-stage
2. stage
3. adminserver-script
4. nameserver-script
5. database-script
6. appserver-script
7. webspeed-script
8. tomcat-script
9. daemon-script
10. mongodb-script
11. mongodbreplica-update-scripts
12. qxo-services-script
13. script-update
14. script-master-update
15. path-fin-dirs-update
16. path-foundation-appserver-update
17. path-foundation-client-update
18. path-foundation-code-update
...
```

update

The *update* command applies configuration changes to the system.

```
> yab update
```

validate

The *validate* command validates the configuration.

```
> yab validate
copy.qadcss-context.source:
[/dr01/build/catalog/packages/yab-css/1/1/0/8/etc/context.xml] is not a directory.
copy.qadcss-verisign.source:
[/dr01/build/work/system/obj/com.qad.yab.css.CssPackageStageProcess/mfgpro/eB3/veris
ign.ini] is not a directory.
copy.qadcss-web.source: [/dr01/build/catalog/packages/yab-css/1/1/0/8/etc/web.xml]
is not a directory.
```

If everything is OK, nothing will be printed to the console.

System Configuration

System Configuration includes topics related to the configuration/reconfiguration of an environment. Configuration changes are applied following the Configuration Change Procedure.

Configuration Change Procedure

Step 1: Define configuration settings in 'configuration.properties'.

`build/config/configuration.properties`

This file only contains settings which differ from the base product configuration (build/config/system) and may initially be empty. To override a setting add the setting to this file and adjust the value or simply adjust the value if the setting is already defined in the file.

Step 2: Execute the update command to apply the configuration change.

Execute the *update* command.

```
> yab update
```

The *update* command can be used to apply any change to the system.

Configuration changes which are limited to updating application configuration files and database settings can be applied with the quicker reconfigure command.

```
> yab reconfigure
```

> The *reconfigure* command does not restart the environment. It may be necessary to restart the environment or specific servers for the application to recognize the change.

Configuration Type System

The resources managed in the QAD Enterprise Applications are associated with configuration settings that are organized using the following pattern:

```
[TYPE].[INSTANCE].[NAME]
```

Where the TYPE designates the type of resource (e.g. "db" for a database) and the INSTANCE designates a specific database (e.g. qaddb). The NAME(s) then are settings to configure that specific instance. Many of these instances "inherit" settings from a prototype instance (_base):

```
[TYPE]._base.[NAME]
```

Settings configured on a prototype will be inherited by all instances of the type (unless the instance defines its own setting).

Types

Type	Description	Instances
adminserver	Progress Admin Server	NO
aia	Application Internet Adapter	YES
appserver	Application Server	YES
code	Code	YES
data	Data	YES
db	Database	YES
dbserver	Database Server	YES
ns	Name Server	NO
packages	Packages	YES
process	Commands	YES
schema	Schema	YES
tomcat	Tomcat	YES
webapp	Web Application	YES
webspeedcode	WebSpeed Code	YES
ws	WebSpeed Server	YES
wsa	Web Services Adapter	YES

Reusable Settings

A convenient way to reuse configuration settings or to temporarily apply and then remove configuration settings is to define the settings in a file that can be copied (and removed) from the system configuration documents directory:

`build/config/system`

The settings can then be applied to the environment with the *update* command.

```
> yab update
```

See the config.order to adjust the precedence order of system documents.

Reconfiguring Databases

Databases are typically associated with two types of configuration settings:

Configure the database.

`db.INSTANCE.*`

Ex.

`db.qaddb.*`

And optionally configuration for a database server.

`dbserver.INSTANCE.*`

Ex.

`dbserver.qaddb.*`

Configuring Database Location

By default all databases will be located in the directory:

`databases`

This default location can be reconfigured with the setting:

`db._base.dir=[DIRECTORY]`

Alternatively to configure the location of a specific database:

`db.[INSTANCE].dir=[DIRECTORY]`

Ex.

```
db.qaddb.dir=/dr01/database
```

> The location of the database should be configured before the environment is created (see QAD Enterprise Edition Installation Guide for details on providing install properties to the YAB Installer.) To change the location of an existing database, the database must either be rebuilt using the command *db-[INSTANCE]-rebuild* or repaired using the Progress *prstrct* tool.

Configuring Database Structure

A structure description (.st) file defines the structure of the database. The .st file is a text file that is used to define t he areas and extents of the database. For detailed information about structure description files, see *OpenEdge Data Management: Database Management*.

To see the structure files used for each database:

```
>yab config "db.*.structurefile"
db.qadadm.structurefile=/dr01/qadapps/qea/build/work/st-files/admd
b.st
db.qadcpl.structurefile=/dr01/qadapps/qea/build/work/st-files/cpld
b.st
db.qaddb.structurefile=/dr01/qadapps/qea/build/work/st-files/mfgdb
.st
db.qadhlp.structurefile=/dr01/qadapps/qea/build/work/st-files/hlpd
b.st
db.qadmodule.structurefile=/dr01/qadapps/qea/build/work/st-files/m
fgdb.st
db.qadrcode.structurefile=/dr01/qadapps/qea/build/work/st-files/rc
ddb.st
db.qxodb.structurefile=/dr01/qadapps/qea/build/work/st-files/qxodb
.st
```

To configure the structure of a specific database:

`db.[INSTANCE].structurefile=[FILE]`

The database must define an AUDIT_DATA and AUDIT_IDX area if it will be enabled for auditing. It is highly recommended that the TABLE, IDX, and LOB area names be retained.

The structure of an existing database can be displayed with the command.

`database-[INSTANCE]-structure-list`

Ex.

```
> yab database-qaddb-structure-list
Area Name: Control Area, Type 6, Block Size 8192, Extents 1, Records/Block 64,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 640 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb.db
Area Name: Primary Recovery Area, Type 3, Block Size 16384, Extents 1
   Ext # 1, Type VARIABLE, Size 65792 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb.b1
Area Name: Schema Area, Type 6, Block Size 8192, Extents 1, Records/Block 64,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 29696 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb.d1
Area Name: MFG, Type 6, Block Size 8192, Extents 2, Records/Block 64, Cluster Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_7.d1
   Ext # 2, Type VARIABLE, Size 1266304 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_7.d2
Area Name: MFG_IDX, Type 6, Block Size 8192, Extents 2, Records/Block 1, Cluster
Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_8.d1
   Ext # 2, Type VARIABLE, Size 724096 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_8.d2
Area Name: AIM, Type 6, Block Size 8192, Extents 2, Records/Block 64, Cluster Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_9.d1
   Ext # 2, Type VARIABLE, Size 6784 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_9.d2
Area Name: AIM_IDX, Type 6, Block Size 8192, Extents 2, Records/Block 1, Cluster
Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_10.d1
   Ext # 2, Type VARIABLE, Size 33408 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_10.d2
Area Name: FIN, Type 6, Block Size 8192, Extents 2, Records/Block 64, Cluster Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_11.d1
   Ext # 2, Type VARIABLE, Size 1153664 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_11.d2
Area Name: FIN_IDX, Type 6, Block Size 8192, Extents 2, Records/Block 1, Cluster
Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_12.d1
   Ext # 2, Type VARIABLE, Size 460416 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_12.d2
Area Name: FIN_LOB, Type 6, Block Size 8192, Extents 2, Records/Block 1, Cluster
Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_73.d1
   Ext # 2, Type VARIABLE, Size 128 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_73.d2
Area Name: AUDIT_DATA, Type 6, Block Size 8192, Extents 1, Records/Block 64, Cluster
Size 8
   Ext # 1, Type VARIABLE, Size 128 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_80.d1
Area Name: AUDIT_IDX, Type 6, Block Size 8192, Extents 1, Records/Block 1, Cluster
Size 8
   Ext # 1, Type VARIABLE, Size 128 KByte, Name:
/dr01/qadapps/qea/databases/mfgdb_81.d1
```

Configuring 4GL Broker Network Support

To enable the 4GL broker to service connections over a TCP/IP network enable the broker's *network* setting to assign the broker a port from the port range range.

Ex. Enable network connections to the qaddb database.

```
dbserver.qaddb.fourgl.network=true
```

As with any port you may explicitly assign the broker port.

Ex. Assign the 4GL broker for the qaddb database a specific port.

```
dbserver.qaddb.fourgl.port=23500
```

By default the application will internally use shared memory connections (for performance reasons) even when a 4GL broker is configured. This can be changed with the following setting.

```
# Configures whether application components should connect to
databases
# over the network or use shared memory. The usual limitations
apply, shared
# memory connections only work when components are deployed on the
same host
# and network connections require the database server to be
configured to
# accept network connections.
progress.database.network=true
```

Configuring a SQL-92 Secondary Login Broker

To enable the secondary broker for a specific database use the setting:

`dbserver.[INSTANCE].sql.network=true`

Ex.

```
dbserver.qaddb.sql.network=true
```

Configuring Before Imaging

To enable database before-imaging on a database:

`dbserver.INSTANCE.beforeimageprocess=true`

Ex.

```
dbserver.qaddb.beforeimageprocess=true
```

Configuring After Imaging

Overview

After-imaging feature lets you recover a database that was damaged when a failure caused the loss of the database or primary recovery (before image) area. When you enable after-imaging, the database engine writes notes containing a description of all database changes to the after-image (AI) files. You can use the AI files with the roll-forward recovery process to restore the database to the condition it was in before you lost the database, without losing completed transactions that occurred since the last backup.

The after image archiver is used to manage after-image extents following Progress best practices. The utility has three major goals:

- Archive FULL AI extents to a user-specified location
- Maintain a log file to aid in ROLL FORWARD
- Be tightly integrated with OpenEdge Replication

Database Structure

After image database extents must be defined before enabling after imaging.

Define the required ai schema areas in new temporary structure file.

Ex. qadadm-ai.st

```
a /dr01/qadapps/qea/databases/admdb.a1
a /dr01/qadapps/qea/databases/admdb.a1
```

Follow the standard Progress steps for adding the qadadm-ai.st structure file to the database. See Progress documentation on "BASIC GUIDE TO AFTER-IMAGING".

Determine the current structure file being used for the database.

```
>yab config db.qadadm.structurefile
db.qadadm.structurefile=/dr01/qadapps/qea/build/work/st-files/admdb.st
```

Copy the current structure file identified above to a new location.

```
> mkdir ai
> cp /dr01/qadapps/qea/build/work/st-files/admdb.st ai/qadadm.st
```

Update *ai/qadadm.st* - adding the ai extent definitions. The merged file may look as follows:

```
#
b /dr01/qadapps/qea/databases
#
d "Schema Area":6,32 /dr01/qadapps/qea/databases
#
d "ADMIN":7,64;8 /dr01/qadapps/qea/databases f 5120
d "ADMIN":7,64;8 /dr01/qadapps/qea/databases
#
d "ADMIN_IDX":8,1;8 /dr01/qadapps/qea/databases f 5120
d "ADMIN_IDX":8,1;8 /dr01/qadapps/qea/databases
#
d "ADMIN_LOB":9,64;8 /dr01/qadapps/qea/databases f 5120
d "ADMIN_LOB":9,64;8 /dr01/qadapps/qea/databases
#
a /dr01/qadapps/qea/databases/admdb.a1
a /dr01/qadapps/qea/databases/admdb.a1
```

Update the *build/config/configuration.properties* to reference this new structure file.

```
db.qadadm.structurefile=/dr01/qadapps/qea/ai/qadadm.st
```

Validate ai extents are available

```
> yab database-qadadm-structure-list
Area Name: Control Area, Type 6, Block Size 8192, Extents 1, Records/Block 64,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 640 KByte, Name:
/dr01/qadapps/qea/databases/admdb.db
Area Name: Primary Recovery Area, Type 3, Block Size 16384, Extents 1
   Ext # 1, Type VARIABLE, Size 16 KByte, Name: /dr01/qadapps/qea/databases/admdb.b1
Area Name: Schema Area, Type 6, Block Size 8192, Extents 1, Records/Block 64,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 3584 KByte, Name:
/dr01/qadapps/qea/databases/admdb.d1
Area Name: ADMIN, Type 6, Block Size 8192, Extents 2, Records/Block 64, Cluster Size
8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/admdb_7.d1
   Ext # 2, Type VARIABLE, Size 48256 KByte, Name:
/dr01/qadapps/qea/databases/admdb_7.d2
Area Name: ADMIN_IDX, Type 6, Block Size 8192, Extents 2, Records/Block 1, Cluster
Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/admdb_8.d1
   Ext # 2, Type VARIABLE, Size 31872 KByte, Name:
/dr01/qadapps/qea/databases/admdb_8.d2
Area Name: ADMIN_LOB, Type 6, Block Size 8192, Extents 2, Records/Block 64, Cluster
Size 8
   Ext # 1, Type FIXED   , Size 5120 KByte, Name:
/dr01/qadapps/qea/databases/admdb_9.d1
   Ext # 2, Type VARIABLE, Size 60032 KByte, Name:
/dr01/qadapps/qea/databases/admdb_9.d2
Area Name: After Image Area 1, Type 7, Block Size 16384, Extents 1
   Ext # 1, Type VARIABLE, Size 256 KByte, Name:
/dr01/qadapps/qea/databases/admdb.a1
Area Name: After Image Area 2, Type 7, Block Size 16384, Extents 1
   Ext # 1, Type VARIABLE, Size 256 KByte, Name:
/dr01/qadapps/qea/databases/admdb.a2
Area Name: AUDIT_DATA, Type 6, Block Size 8192, Extents 1, Records/Block 64, Cluster
Size 8
   Ext # 1, Type VARIABLE, Size 128 KByte, Name:
/dr01/qadapps/qea/databases/admdb_80.d1
Area Name: AUDIT_IDX, Type 6, Block Size 8192, Extents 1, Records/Block 1, Cluster
Size 8
   Ext # 1, Type VARIABLE, Size 128 KByte, Name:
/dr01/qadapps/qea/databases/admdb_81.d1
```

Enabling After Imaging

If the database is offline, you must have a recent backup of the database. If the database has changed since the last backup you will be prompted to backup the database first. If the database is online a timestamped backup will automatically be created.

To enable AI on all databases, execute the command:

```
> yab database-ai-enable
```

Alternatively to enable AI on a specific database, execute the command:

```
> yab database-[INSTANCE]-ai-enable
```

Configuring After Image Writer

Configuring an after-image writer (AIW) will reduce the I/O required to support after-imaging. The AIW will be started and stopped automatically when the database is started and stopped.

`dbserver.[INSTANCE].afterimageprocess=true`

Ex.

```
dbserver.qaddb.afterimageprocess=true
```

Configuring AI Archiver

The *archivaldir* setting configures the location where AI files are written.

`dbserver.[INSTANCE].archivaldir=[DIRECTORY]`

The default setting writes the AI files to:

`build/work/ai`

Enabling AI archiver

The AI archiver must be enabled when the target databases are offline. Once enabled the archiver process will automatically start and stop when the database is started and stopped.

To enable the AI archiver on all databases, execute the command:

```
> yab database-ai-archiver-enable
```

Alternatively to enable the AI archiver on a specific database, execute the command:

```
> yab database-[INSTANCE]-ai-archiver-enable
```

Configuring Codepage and Collation

The factory default settings create UTF-8 databases using the ICU-UCA collation.

The codepage and collation can be configured for all databases using the following set of properties:

```
# The codepage of the database.
db._base.codepage=

# A directory under '$DLC/prolang' that is used when creating the
# database and to locate collation data.
db._base.template=

# The collation of the database.
db._base.collation=
```

Ex. ISO8859-1 with Basic collation and American English (ame) collation.

```
db._base.codepage=ISO8859-1
db._base.template=ame
db._base.collation=basic
```

Using Demo Data

QAD provides demo data for demonstrating and testing application features. In a non-production environment, the demo data can be temporarily loaded into an environment following these steps.

Check Currently Configured Data.

Before changing the dataset in the environment, first check the current dataset that is installed to verify an update is required:

```
> yab config packages.ee-data
packages.ee-data=release-data-ee2016-2016.0.16.209
```

Loading Demo Data

Backup Databases

Create a backup of all databases. The tag "initial-backup" can be replaced with a more meaningful name.

```
> yab -tag:initial-backup database-backup
```

Configure Demo Data

The YAB installer is used to reconfigure the environment to use the demo data.

```
> yab install /dr01/installs/image -install-update:false
```

Proceed through the installer prompts up to the 'Feature Selection' page, and enter '3' to select the Demo Data.

```
[Feature Selection]
Choose the features to install:
 [X] 1 - Enterprise Edition
 [X] 2 - QXtend
 [X] 3 - Demo Data
Hit ENTER to move to the next question.
```

Start the installation to reconfigure the environment.

```
[Start Installation]
Start installing QAD Enterprise Edition 2016?
 n - No
 y - Yes
> y

Validating
OK
Initializing instance
OK
Reconfiguring instance
OK
Resolving and copying packages
OK
Cleaning up
OK
INSTALL SUCCESSFUL
```

Rebuild Databases & Update

```
> yab database-rebuild
> yab -clean update
```

If you plan on switching between standard and demo data frequently you can create a backup of the databases with the demo data loaded and replace the *database-rebuild* command in the previous step with the *database-restore* command (referencing the appropriate tag).

```
> yab -tag:demo-data database-backup
```

Restoring Original Data

To restore the original data.

Stop Environment

```
> yab stop
```

Unconfigure Demo Data

The YAB installer is used to reconfigure the environment to use the release data.

```
> yab install /dr01/installs/image -install-update:false
```

Proceed through the installer prompts up to the 'Feature Selection' page, and ensure '3 - Demo Data' is not selected.

```
[Feature Selection]
Choose the features to install:
 [X] 1 - Enterprise Edition
 [X] 2 - QXtend
 [ ] 3 - Demo Data
Hit ENTER to move to the next question.
```

Start the installation to reconfigure the environment.

```
[Start Installation]
Start installing QAD Enterprise Edition 2016?
 n - No
 y - Yes
> y

Validating
OK
Initializing instance
OK
Reconfiguring instance
OK
Resolving and copying packages
OK
Cleaning up
OK
INSTALL SUCCESSFUL
```

Restore Initial Backup & Update

```
> yab -tag:initial-backup database-restore update
```

Updating PROPATH

Progress sessions use the PROPATH to locate resources. The PROPATH is a list of paths where requests for resources are satisfied by the first path in which the resource is defined. To simplify the management of PROPATH information, QAD Enterprise Applications defines a set of base PROPATH definitions which are then reused in different application sessions (telnet, client, appservers, etc).

Base Definitions

Setting	Description
propath.base	The most fundamental PROPATH definition inherited by many sessions.
propath.mfg	Extends *propath.base* to provide a base PROPATH for operational sessions.
propath.fin	Extends *propath.base* to provide a base PROPATH for Finance sessions.

Analysis

The following command lists all of the PROPATH settings and may be used to determine the setting or setting(s) that control the Progress sessions of interest.

```
> yab -skip-value config "*propath*"
appserver.fin.propath
appserver.mfg.propath
appserver.qra.propath
appserver.qxosi.propath
appserver.qxoui.propath
appserver.qxtnative.propath
clientscript._base.properties.propath
code.fin.propath
...
```

The (-trace) option can be used to drill into a specific PROPATH to see how it was constructed. In the following example we see that the PROPATH that controls the Financials application server inherits the paths from *propath.fin*, which in turn inherits the paths from *propath.base*, and that there are independent contributions to the path.

```
> yab -trace -skip-value config appserver.fin.propath
appserver.fin.propath (inherited: propath.base => propath.fin)
  @ref propath.fin :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s
  @ref propath.base :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s
  ${dist.dir}/fin/patch,${appdir}/dist/fin :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s (+)
  ${packages.fin-bldata.dir},${packages.fin-bin64-qadfin.dir}/qadfin.pl :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s (+)
  ${appdir}/config :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s (+)
  ${netui.pro.dir}/com/mfgpro :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s (+)
  .,${packages.qra-oe11.dir}/lib/qra.pl :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s (+)
  ${code.mfg.dir},${code.mfg.dir}/us,${code.mfg.dir}/us/bbi :
build/config/packages/yab-ee-foundation/1/3/0/50/default/yab-ee-foundation.propertie
s (+)
  ${packages.qracore.dir}/qad.qra.core/bin/qracore.pl :
build/config/packages/yab-qra/1/3/0/25/default/yab-qra.properties (+)
  ${code.qxi.dir} :
build/config/packages/yab-qxtend/1/3/0/27/default/yab-qxtend.properties (+)
```

Adding

To add a path to a PROPATH choose one of the following options:

Standard Priority

```
# @id [PATH ID]
# @append
[SETTING]=[PATH],[PATH]...
```

or

Non-Standard Priority

```
# @id [PATH ID]
# @append
# @group [GROUP ID]
[SETTING]=[PATH],[PATH]...
```

The PATH ID can be used to refer to the path(s) when ordering the PROPATH (see below) and is optional. The GROUP ID is used to set the priority of the added path(s) using the following table:

Group ID	Description
0	Bootstrap
1	Configuration
2	Customizations
3	Patches

(undefined)	Standard

Ex.

```
# @id test
# @append
code.mfg.propath=/test/programs
```

Ordering

The group is used to define a first level ordering of paths. YAB guarantees that a path will be preceded by all paths associated with a lower group index and followed by all paths associated with a higher group index. Paths associated with the same group have an indeterminate order unless the Group Order annotation is used to define a secondary ordering of the members of a group.

Group Order

To order the members of a group choose one of the following options.

Standard Priority

```
# @grouporder [PATH ID] [PATH ID]...
[SETTING]
```

or

Non-Standard Priority

```
# @grouporder [GROUP ID] [PATH ID] [PATH ID]...
[SETTING]
```

Paths not included in the list will be ordered after the listed members in an indeterminate order.

Ex.

```
# @grouporder test main
code.mfg.propath=
```

Migrating to a New Host

The *host* setting is initialized to the host where the instance was built.

```
> yab config host
host=vmlinux.qad.com
```

The term 'instance' refers to the installed environment. In this guide the instance is wholly contained in /dr01/qadapps/qea. Depending on configuration options the instance may use different filesystem locations for ex. databases. In such a case all application components that compose the instance would need to be moved.

An instance can be migrated to a new host if the following preconditions are met:

- The instance will be located in the same directory on the target host.
- The source and target host run the same operating system.

Step 1: Stop the environment (and any manually controlled servers) on the old host.

```
> yab stop
```

Step 2: Configure the new host.

```
host=[NEW HOST]
```

Step 3: Move instance files to the new host.

If the instance files are on a local file system.

Step 4: Update and restart the environment on the new host.

```
> yab update restart
```

Reconfiguring Tomcat

Enabling the Manager Web Application

The Tomcat Manager web application is distributed with Tomcat, but for security reasons, no accounts are configured to allow access to the manager by default. The "manager-gui" role is a Tomcat role with access to the manager's HTML interface.

To provide access to the Manager web application with the username (admin) and the password (mfgpro).

```
tomcat.[INSTANCE].roles.manager-gui.rolename=manager-gui
tomcat.[INSTANCE].roles.manager-gui.members=admin
tomcat.[INSTANCE].users.admin.username=admin
tomcat.[INSTANCE].users.admin.password=mfgpro
```

To add additional users...

```
tomcat.[INSTANCE].users.[USER].username=[USER]
tomcat.[INSTANCE].users.[USER].password=
```

To add additional roles...

```
tomcat.[INSTANCE].roles.[ROLE].rolename=[ROLE]
tomcat.[INSTANCE].roles.[ROLE].members=[COMMA-SEPARATED USERS]
```

In the preceding example it is the recommended convention to use the user's name and the role name when defining the setting:

`tomcat.[INSTANCE].users.[USER].username=[USER]`

`tomcat.[INSTANCE].roles.[ROLE].rolename=[ROLE]`

But the only requirement is that each user and role has a unique instance name:

`tomcat.[INSTANCE].users.[UNIQUE USER ID].username=[USER]`

`tomcat.[INSTANCE].roles.[UNIQUE ROLE ID].rolename=[ROLE]`

Reconfiguring Tomcat Startup Parameters

The Java startup parameters for Tomcat are defined by the *startupopts* setting. To change the startup parameters that are inherited by all Tomcat servers.

```
tomcat._base.startupopts="-XX:MaxPermSize=256m"
```

Alternatively to change the startup parameters of a specific Tomcat server.

```
tomcat.[INSTANCE].startupopts="-XX:MaxPermSize=512m"
```

Configuring SSL Connections

Tomcat can be configured to support SSL connections. The configuration of SSL requires an X.509 certificate that digitally binds a cryptographic key to an organization's details. The certificate must be imported into the *keystore* th at is associated with the Tomcat instance to be secured.

The default configuration associates all Tomcat instances with the keystore:

`build/work/keystore/default.bin`

Ex. Show the keystore that the default Tomcat instance is associated with.

```
> yab config tomcat.default.keystore
tomcat.default.keystore=/dr01/qadapps/qea/build/work/keystore/default.bin
```

Ex. Show the configuration of the default keystore.

```
> yab config keystore.default.*
keystore.default.file=/dr01/qadapps/qea/build/work/keystore/default.bin
keystore.default.password=changeit
```

> The keystore is created when the first certificate is imported.

Change the Keystore Password

The keystore password should be set/changed before importing certificates into the keystore. If the password is changed after the certificates are imported the certificates will need to be imported again after changing the password.

```
keystore.default.password=abetterpassword
```

Remove the existing keystore. If it does not exist, nothing will be done.

```
> yab keystore-default-remove
```

> The keystore password is stored in the clear in the *configuration.properties* file and in the *conf/server.xml* file associated with all configured Tomcat instances. These files should have appropriate OS permissions set.

Import Certificates

To import a certificate use either of the following commands. An alias can be used to uniquely identify the certificate if multiple certificates will be imported into the keystore (Tomcat will use the first certificate in the keystore unless configured with an alias to use).

`yab keystore-[INSTANCE]-import -key:[PRIVATE KEY] -certificate:[CERTIFICATE CHAIN]`

or

```
yab keystore-[INSTANCE]-import -key:[PRIVATE KEY] -certificate:[CERTIFICATE CHAIN]
-alias:cert1
```

More information is available in the command help:

```
> yab help keystore-default-import
PROCESS
     keystore-default-import - Imports a certificate into the 'default' keystore.
DESCRIPTION
     If the alias is already defined in the keystore, the prior entries will be
replaced.
     The openssl toolkit can be used to create a self-signed certificate for
testing:
     openssl req -x509 -newkey rsa:2048 -days 10000 -nodes -keyout key.pem -out
cert.pem
     and can be used to convert keys into the format required by this API:
     openssl pkcs8 -topk8 -in key.pem -inform pem -out key.der -outform der -nocrypt
OPTIONS
     Name            Description
     -----           -----------
     alias           The alias to identify the key in the keystore.
                     If not defined the alias 'default' will be used.
     key             The DER encoded private key file in PKCS#8 format (required).
     certificate     The DER encoded certificate file in X.509 format (required).
```

The following command will display the certificates in a keystore:

`yab keystore-[INSTANCE]-print`

Reconfigure Tomcat

Enable SSL on the desired Tomcat instances and the apply the changes. If the Tomcat instances are running they will need to be restarted for the change to take effect.

Ex. Configure the default Tomcat instance to use SSL.

```
tomcat.default.usessl=true
tomcat.default.keyalias=cert1 //if multiple certificates are
imported
```

Execute the following command to apply the change.

`yab tomcat-[INSTANCE]-update`

Ex. Update the default Tomcat instance.

```
> yab tomcat-default-update
```

To display the TCP/IP port used for SSL connections:

```
> yab config tomcat.default.sslport
tomcat.default.sslport=22014
```

TCP/IP Ports

Servers use TCP/IP ports for communication.

Static Ports

Static TCP/IP ports can be configured like any other setting.

Ex. Configure the Progress Admin Server to use port 22001.

```
# @port
adminserver.port=22001
```

Alternatively the *ports* setting can define a range of ports from which ports are automatically allocated:

`ports=[LOWER BOUND]-[UPPER BOUND]`

Ex.

```
ports=28000-28100
```

If the *ports* setting is not defined it defaults to the range 22000-22100.

The statement (# @port) in the example above is a port annotation. When a port is explicitly assigned a value, the annotation simply serves to let the system know that the port has been assigned and should not be allocated. When the annotation is used without specifying a value, an available port will be allocated from the range.

```
# @port
adminserver.port=
```

After a port is allocated to a setting, the port will continue to be allocated to that setting, unless explicitly overridden. In the following example, the annotation requests a specific port from the range using an offset, where 0 is associated with the first port in the range, 1 the second, and so on. If the requested port is already assigned, an available port from the range will be allocated.

Ex. Configures a port using an offset.

```
# @port +0
adminserver.port=
```

Dynamic Ports

Similar to the ports range, a range of dynamic ports can be allocated with the same pattern

`dynamic-ports=[LOWER BOUND]-[UPPER BOUND]`

Ex.

```
> yab config dynamic-ports*
dynamic-ports=1026-2000
```

This dynamic-ports range is used to allocate a port range for servers that automatically choose an available TCP/IP port each time they are started.

These ports can also be configured directly as outlined below.

Server	Range (Start)	Range (End)
Webspeed	ws.[INSTANCE].srvrminport	ws.[INSTANCE].srvrmaxport
Application Server	appserver.[INSTANCE].srvrminport	appserver.[INSTANCE].srvrmaxport
Database Server (4GL)	dbserver.[INSTANCE].fourgl.maxdynamicport	dbserver.[INSTANCE].fourgl.maxdynamicport
Database Server (SQL Broker)	dbserver.[INSTANCE].fourgl.mindynamicport	dbserver.[INSTANCE].sql.maxdynamicport

Adding Languages

Configure Language

To add an additional language adjust the *languages* setting:

```
# A comma-delimited list of the languages to support.
# Available: us,ch,cs,cz,du,fr,ge,it,jp,ko,ls,pl,po,tw
languages=us,ge
```

Non Core Languages

QAD Enterprise Applications distributes resources for the following core languages:

Code	Language
us	English
ch	Simplified Chinese
cs	Spanish
cz	Czech
du	Dutch
fr	French
ge	German
it	Italian
jp	Japanese
ko	Korean
ls	Spanish (Mexico)
pl	Polish
po	Portuguese (Brazil)
tw	Traditional Chinese

If the language being added is not one of the core languages, you must also configure the Financials patch with support for the language as well as the language specific *qad-lang* image that provides the translations.

See documentation included in the *qad-lang* image for additional details.

Apply Changes

Run an update to add the language.

```
> yab update
```

Use AIA

To configure the .NET client to send requests through a Progress Application Internet Adapter:

Step #1: Define the setting

```
netui.aia.enabled=true
```

Step #2: Apply the setting

```
> yab stop webapp-homeserver-client-session-update start
```

The Financials client components are not currently compatible with AIA and will continue to connect outside of an AIA adapter.

To enable HTTPS on the default Application Internet Adapter:

```
aia.default.httpsenabled=true
```

AdminServer

The Progress AdminServer is configured through the *adminserver* type.

```
> yab help adminserver.
 adminserver.adminport                    The TCP/IP port used for communication
between the admin server and databases.
 adminserver.conmgr                       Locates the connection manager
configuration file.
 adminserver.plugins                      Locates the plugin configuration file.
...
```

Properties in the plugin configuration file can be defined using the following pattern:

`adminserver.plugins.[INSTANCE].section=[section to modify]`

`adminserver.plugins.[INSTANCE].property=[property]`

`adminserver.plugins.[INSTANCE].value=[value]`

Ex. Define the *jvmargs* plugin setting.

```
adminserver.plugins.jvmargs.section=PluginPolicy.Progress.AdminSer
ver
adminserver.plugins.jvmargs.property=jvmargs
adminserver.plugins.jvmargs.value=-Xmx256m
-Djava.awt.headless=true
-Dsun.lang.ClassLoader.allowArraySyntax=true
-Dserver.start.retryInterval=2
```

Result:

```
[PluginPolicy.Progress.AdminServer]
 jvmargs=-Xmx256m -Djava.awt.headless=true
-Dsun.lang.ClassLoader.allowArraySyntax=true
-Dserver.start.retryInterval=2
```

Customizing YAB

The *process* setting is used to reconfigure existing commands and to hook new commands into the environment. The setting is an instanced setting using the following pattern.

`process.[INSTANCE].[NAME]`

Ex. Define a new command *compile* to recompile the operational code and then trim the operational application server.

```
process.compile.id=compile
process.compile.depends=code-mfg-update,appserver-mfg-trim
```

```
> yab system-process-list code-mfg-update
yab system-process-list code-mfg-update
1. module-archiving-stage
2. code-mfg-update
```

```
> yab system-process-list compile
1. module-avataxeeconnector-stage
2. module-label-print-stage
3. module-data-collections-unencrypted-stage
4. code-mfg-update
5. appserver-mfg-trim
6. compile
```

The system processes associated with a given process depend on the installed modules.

See Also:

Shell Script

ANT Script

Shell Script

To define a command that executes an OS command the *impl* setting is set to the value "exec".

Ex. Define a command 'greeting' that executes a shell script of the same name.

```
process.greeting.id=greeting
process.greeting.args=/dr01/qad-scripts/greeting
process.greeting.impl=exec
```

Ex. Pass an argument to the script.

```
process.greeting.args=/dr01/qad-scripts/greeting ${appdir}
```

or

```
process.greeting.args.0=/dr01/qad-scripts/greeting
process.greeting.args.1=${appdir}
```

Ex. Define an environment variable for the script.

```
process.greeting.env.APPDIR=${appdir}
```

Ex. Configure the greeting to be executed after the environment is started.

```
process.start.id=start
process.start.anchorafter=greeting
```

Ex. A contrived example exposing the long form of the "ls" command as the process "ll".

```
process.ls.id=ll
process.ls.args=ls -l
process.ls.impl=exec
process.ls.console=true
process.ls.status=false
process.ls.greedy=true
```

Using the process...

```
> yab -F ll build
total 16
drwxrwxr-x 37 mfg qad 4096 Mar  3 12:17 work/
drwxrwxr-x  5 mfg qad 4096 Mar  6 14:21 logs/
drwxrwxr-x  6 mfg qad 4096 Mar  7 06:05 config/
drwxrwxr-x  3 mfg qad 4096 Feb 24 08:07 catalog/

BUILD SUCCESSFUL (3.545 s)
```

To print help for the *process* configuration type.

```
> yab help process.
```

ANT Script

To define a command that executes an Apache ANT script the *impl* setting is set to the value "ant".

Ex. Define a command 'ant-test' that executes the 'test' target of the ANT script

```
process.ant-test.id=ant-test
process.ant-test.summary=A test to execute an ANT script.
process.ant-test.impl=ant
process.ant-test.args.0=-script:/dr01/qad-scripts/build.xml
process.ant-test.args.1=-targets:test
```

If we wanted the *ant-test* process to run whenever a user runs an update we could add the following settings.

```
process.update.id=update
process.update.anchorbefore=ant-test
```

Within the ANT script all application properties are mapped to ANT properties.

```
<project>
    <target name="test">
  <echo level="info">Current Environment: ${environment.id}</echo>
    </target>
</project>
```

```
> yab -v ant-test
...
2014-09-10 11:15:55,834 DEBUG BuildContext - ant-test
2014-09-10 11:15:55,845 DEBUG Ant - Script:
/dr01/qadapps/qea/build/catalog/packages/ant-test/1/0/0/0/build.xml
2014-09-10 11:15:55,845 DEBUG Ant - Targets: [test]
2014-09-10 11:15:55,845 DEBUG Ant - Using script file:
/dr01/qadapps/qea/build/catalog/packages/ant-test/1/0/0/0/build.xml
2014-09-10 11:15:56,355 INFO  Echo - Current Environment: test
2014-09-10 11:15:56,355 DEBUG BuildContext - ant-test OK
```

Key Settings

catalogs setting

Configures a comma-delimited list of software catalogs from highest to lowest precedence, where each entry may be a file system path or URL locating the catalog.

`catalogs=[FILE|URL],[FILE|URL]...`

The local software catalog should not be configured.

Ex.

```
catalogs=/shared/cache,http://packages.qad.com
```

check-for-updates setting

Controls whether package configuration settings are automatically re-evaluated when the configuration changes. A value of false skips the evaluation of packages.

```
check-for-updates=false
```

When *check-for-updates* is configured as false, the setting can be temporarily overridden to process and apply package changes.

```
> yab -check-for-updates update
```

The *check-for-updates* setting is designed for production environments to strictly control the interaction between the instance and the configured catalogs setting.

Default: true

clean setting

Forces the system to avoid using any cached information to process the request.

This is typically used as a command line option as necessary.

```
yab -clean ...
```

config.order setting

Defines the precedence order of configuration documents in the directory:

`build/config/system`

Documents are listed from highest precedence to lowest precedence:

Ex. Settings in FILE1 take precedence over FILE2.

`config.order=[FILE1],[FILE2]...`

To show the existing order:

```
> yab config config.order
```

You can adjust the order by editing the setting.

dependency settings

A family of settings to fine tune the evaluation of package settings.

dependency.evaluation

Configures a strategy for choosing an optimal solution when the evaluation of package settings can be satisfied by multiple solutions.

```
dependency.evaluation=[HighestVersion|LeastChange]
```

Default: HighestVersion
HighestVersion

Prefer solutions that include the highest versions of packages. (Default)
LeastChange

Prefer solutions that include packages that have already been applied to the application.

Ex. Configure the LeastChange evaluation strategy.

```
dependency.evaluation=LeastChange
```

dependency.excludes

Configures a comma-delimited list of package ranges that should be excluded from the solution.

```
dependency.excludes=[PACKAGE RANGE],[PACKAGE RANGE]...
```

Ex. Exclude all 'abc' 1.x packages from the solution.

```
dependency.excludes=abc[1,2)
```

dependency.excludesreferences

Configures a comma-delimited list of package ranges whose dependencies should be excluded/ignored.

```
dependency.excludesreferences=[PACKAGE RANGE],[PACKAGE RANGE]...
```

Ex. Excludes the dependencies of all abc 1.x packages from the evaluated solution.

```
dependency.excludesreferences=abc[1,2)
```

dependency.overrides

Configures a comma-delimited list of package ranges whose dependencies should be overridden.

```
dependency.overrides=[PACKAGE RANGE]=[DEPENDENCY STATEMENT],[PACKAGE
RANGE]=[DEPENDENCY STATEMENT]...
```

Ex. Replaces the dependency information associated with abc 1.x packages with a dependency on any version of the xyz package.

```
dependency.overrides=abc[1,2)=xyz
```

dependency.redirects

Configures a comma-delimited list of dependency redirections.

```
dependency.redirects=[PACKAGE RANGE]=[PACKAGE NAME],[PACKAGE RANGE]=[PACKAGE NAME]...
```

Ex. Restate all references to 1.x versions of the abc package as references to version 3.4.12 of the abc package.

```
dependency.redirects=abc[1,2)=abc-3.4.12
```

packages setting

Configures packages in the system using the syntax:

```
packages.NAME=VALUE
```

NAME

The NAME is an identifier that is used internally to refer to the package. Typically the NAME must be identical to the package name.

Ex.

```
packages.fin-src-proxy=fin-src-proxy-2015.0.80.11
```

> In special cases the NAME is different than the package name. This is best illustrated with an example. There are several packages that distribute data to initialize a system. In YAB each of these variants is associated with the NAME *ee-data* (to provide the management components with a consistent "handle" to reference the data packages).
>
> Ex.
>
> ```
> packages.ee-data=release-data-ee2016-2016.0.16.209
> packages.ee-data=system-test-data-ee2016-2016.0.16.209
>
> packages.ee-data=demo-data-ee2016-2016.0.16.209
> ```

VALUE

The VALUE is used to define a range of acceptable versions for the package. Typically the VALUE configures a specific version. When the NAME and the package name are identical, the package name may be omitted from the VALUE.

Ex.

```
packages.fin-src-proxy=2016.0.80.5
```

Here are some examples where a range is configured using the *netui-module-dashboard-cm* package for illustration:

Accepts any version

```
packages.netui-module-dashboard-cm=
```

Accepts any version >= 1.0

```
packages.netui-module-dashboard-cm=[1]
```

Accepts any version >= 1.0 and < 2.0

```
packages.netui-module-dashboard-cm=[1,2)
```

Conditional Package Inclusion / Exclusion (if / unless)

A package can be conditionally configured so that whether or not it is enabled depends on whether or not other packages are configured (if condition) or not configured (unless condition).

Syntax:

```
packages.NAME.if=PACKAGE CLASS, PACKAGE CLASS...
packages.NAME.unless=PACKAGE CLASS, PACKAGE CLASS...
```

Ex. Enable the 'yab-event-service' package if the 'event-service' package is configured.

```
packages.yab-event-service=1.3.0.12
packages.yab-event-service.if=event-service
```

Ex. Enable the 'mongodb' package if either the 'event-service' or 'collaboration' package is configured.

```
packages.mongodb=mongodb-rhel5-3.0.3.0
packages.mongodb.if=event-service,colloboration
```

> The package conditions are evaluated before packages are resolved.

ports setting

Configures a range of TCP/IP ports to allocate to services that do not have a fixed port.

`ports=[LOWER BOUND (inclusive)]-[UPPER BOUND (exclusive)]`

Ex. Allocate ports from 16500 up to and including 16549.

```
ports=16500-16550
```

verify setting

Instructs YAB to verify commands after they execute.

A command may be associated with tests that verify the system after the command has executed. For example, the command to start a Tomcat server will submit a request to start the server. This request does not block (wait) and will return immediately if the request was submitted correctly. The verification of this command will block for a predetermined amount of time until the HTTP or HTTPS port is open and all testable web applications are active.

This is typically used as a command line option as necessary.

```
yab -verify start

                            start (34 tasks)
------------------------------------------------------------------------------
1/34  adminserver-start                                   STARTED (6.515 s)
      adminserver-start                                   PASSED  (3.303 s)
2/34  nameserver-start                                    OK (0.000 s)
      nameserver-start                                    PASSED  (38.409 s)
3/34  database-alerts-start                               STARTED (2.711 s)
      database-alerts-start                               PASSED  (0.043 s)
4/34  database-bisgen-start                               STARTED (2.673 s)
      database-bisgen-start                               PASSED  (0.031 s)
5/34  database-bisgmenu-start                             STARTED (2.371 s)
      database-bisgmenu-start                             PASSED  (0.030 s)
6/34  database-configurator-start                         STARTED (2.453 s)
      database-configurator-start                         PASSED  (0.030 s)
7/34  database-dataaccess-start                           STARTED (2.439 s)
      database-dataaccess-start                           PASSED  (0.031 s)
```

An alternative option (-verify-no-exec) will enable verification while preventing any commands from executing. This variant might be used to run the verifications associated with a command without executing the command.

```
yab -verify-no-exec start

                            start (34 tasks)
------------------------------------------------------------------------------
1/34  adminserver-start                                   SKIPPED (0.000 s)
      adminserver-start                                   PASSED  (1.793 s)
2/34  nameserver-start                                    SKIPPED (0.000 s)
      nameserver-start                                    PASSED  (1.097 s)
3/34  database-alerts-start                               SKIPPED (0.000 s)
      database-alerts-start                               PASSED  (0.081 s)
4/34  database-bisgen-start                               SKIPPED (0.000 s)
      database-bisgen-start                               PASSED  (0.043 s)
5/34  database-bisgmenu-start                             SKIPPED (0.000 s)
      database-bisgmenu-start                             PASSED  (0.039 s)
6/34  database-configurator-start                         SKIPPED (0.000 s)
      database-configurator-start                         PASSED  (0.061 s)
7/34  database-dataaccess-start                           SKIPPED (0.000 s)
      database-dataaccess-start                           PASSED  (0.036 s)
```

Upgrades, Customizations, and Patches

Upgrades, Customizations, and Patches describes how to manage the software running in an environment.

Product Upgrades

Overview

Product updates are delivered as a zip file. This zip file contains the packages to be installed in the QAD Enterprise Edition instance. Product updates may provide their own upgrade notes / steps that must be carried out after installation of the product zip. These upgrade notes should be reviewed before upgrading any product.

To install the zip file:

```
> yab install product-x.x.x.x.zip
```

Alternatively to make the upgraded packages available without applying to the environment execute the following:

```
> yab install product-x.x.x.x.zip -install-update:false
```

This allows you examine the environment package versions before applying the change.

To apply the changes

```
> yab update
```

Customizations

Customizations are files that supplement or replace standard product files. The approach to integrating a customization varies according to the application component being customized, but in general customizations are organized in the customizations directory (configured by the setting *customizations.dir*) and then applied to the application with the *update* command.

```
customizations.dir=${appdir}/customizations
```

Operational Customizations

An operational customization can include Progress programs and data. Customizations are organized in the directory configured by the *customizations.mfg.dir* setting, with the default:

```
customizations/mfg
```

Setup

To define a new customization create a sub-directory for the customization:

```
mkdir customizations/mfg/cust1
```

Create directories for the source code (as necessary):

```
mkdir -p customizations/mfg/cust1/src/us/so
mkdir -p customizations/mfg/cust1/src/us/wo
```

Create directories for the data (as necessary):

```
mkdir -p customizations/mfg/cust1/data/qadadm
mkdir -p customizations/mfg/cust1/data/qaddb
```

Copy the source code into the *src* directory and the data (Progress .d or QAD XML) into the *data* directory.

> As a convenience a *default* customization is pre-defined:
>
> ```
> > tree customizations/mfg/default
> customizations/mfg/default/
> |-- data
> | |-- qadadm
> | `-- qaddb
> `-- src
> ```

Installing

To compile the customization execute the *update* command.

```
> yab update
```

The data will be loaded into the relevant databases and the operational code will be recompiled, including the customized sources, into a directory configured by the *code.mfg.dir* setting, with the default:

```
dist/mfg
```

> YAB will automatically order the compile PROPATH so that customizations take precedence over standard programs. If the relative order of customizations is important a *grouporder* annotation may be defined on the *code.mfg.propath* setting to fine-tune the operational code compile PROPATH. Customizations that are not enumerated in the *grouporder* annotation will be ordered after the enumerated customizations.
>
> Ex.
>
> ```
> # @grouporder 2 cust2 cust1
> code.mfg.propath=
> ```

Removing

To remove customized programs from the system (source code and rcode):

```
rm -rf customizations/mfg/[NAME]/*
yab code-mfg-update
```

Financials Customization

The Financials component is customized by defining Progress programs that adhere to the Financials Non-Intrusive Customizations guidelines.

Defining Financial Customizations

Financials Non-Intrusive Customizations templates are distributed in the package `fin-customizations.`

The location of this package on the system can be determined by executing this command.

```
> yab config packages.fin-customization.dir
packages.fin-customization.dir=/dr01/qadapps/qea/build/catalog/packages/fin-customiz
ation/2016/0/80/5
```

Financials customizations should be defined in the location configured by the `customizations.fin.dir` setting.

Execute the following to find the location.

```
> yab config customizations.fin.dir
customizations.fin.dir=/dr01/qadapps/qea/customizations/fin
```

Template files can be copied into the `customizations.fin.dir` from the package directory for development.

Compiling Customization Programs

The customization programs will be compiled into the location specified by the `code.fin.dir` setting.

Execute the following to find the location.

```
> yab config code.fin.dir
code.fin.dir=/dr01/qadapps/qea/dist/fin/customcode
```

This is automatically included in the Financials appserver PROPATH:

```
> yab config appserver.fin.propath
appserver.fin.propath=/dr01/qadapps/qea/config,/dr01/qadapps/qea/b
uild/catalog/packages/qracore/2/17/0/32/qad.qra.core/bin/qracore.p
l,/dr01/qadapps/qea/dist/fin/patch,/dr01/qadapps/qea/dist/fin,
/dr01/qadapps/qea/dist/pro/com/mfgpro,/dr01/qadapps/qea/build/cata
log/packages/fin-bldata/2016/0/80/5,/dr01/qadapps/qea/build/catalo
g/packages/fin-bin64-qadfin/2016/0/80/5/qadfin.pl,.,
/dr01/qadapps/qea/build/catalog/packages/qra-oe11/1/1/166/0/lib/qr
a.pl,/dr01/qadapps/qea/dist/mfg,/dr01/qadapps/qea/dist/mfg/us,/dr0
1/qadapps/qea/dist/mfg/us/bbi,/dr01/qadapps/qea/dist/qxtadpt
```

Applying Financial Customization

To apply Financial Customizations the `update` command must be executed.

```
> yab update
```

Alternatively as a convenience the following command can be executed to apply the customizations when the change is a code change only.

```
> yab code-fin-update
```

Adding Custom Database

Additional databases and database servers can be configured.

Database

Defines a new database named 'custom' that inherits the default database settings.

```
# @extends db._base
db.custom=
db.custom.physicalname=custom
```

To display the configuration of the database.

```
> yab config db.custom.*
db.custom.bi=16384
db.custom.biblocksize=16
db.custom.blocksize=8
db.custom.centuryformat=1950
db.custom.codepage=utf-8
db.custom.collation=ICU-UCA
db.custom.dir=/dr01/qadapps/qea/databases
...
```

To display help for database settings.

```
> yab help db.
```

Database Server

Defines a database server that inherits the default database server settings.

```
# @extends dbserver._base
dbserver.custom=
dbserver.custom.name=db-${db.custom.physicalname}
dbserver.custom.databasename=${db.custom.dir}/${db.custom.physical
name}
dbserver.custom.fourgl.network=true
dbserver.custom.fourgl.port=23600
```

> The database server is associated with the database by sharing the same INSTANCE name. For example, the "dbserver.custom" is the database server for "db.custom" because they share the instance name "custom".

To display the configuration of the database.

```
> yab config dbserver.custom.*
dbserver.custom.afterimagebuffers=50
dbserver.custom.archivaldir=/dr01/qadapps/qea/build/work/ai
dbserver.custom.archivalinterval=120
dbserver.custom.asynchronouspagewriters=1
dbserver.custom.beforeimagebuffers=50
dbserver.custom.blocksindatabasebuffers=1920
dbserver.custom.collationtable=ICU-UCA
dbserver.custom.databasename=/dr01/qadapps/qea/databases/foo
dbserver.custom.fourgl.host=vmwtb01
dbserver.custom.fourgl.maxclientsperserver=10
dbserver.custom.fourgl.minclientsperserver=5
...
```

To display help for database server settings.

```
> yab help dbserver.
```

Use the following setting to define a database server that does not automatically start and stop when the environment is started and stopped.

dbserver.custom.manual=true

Schema

To define schema for the custom database:

```
schema.custom.database=db.custom
schema.custom.file=/dr01/qadapps/qea/customizations/database/custo
m.df
```

Update

To apply the configuration changes execute the *update* command.

```
> yab update
```

Integrated Customization Toolkit

The Integrated Customization Toolkit (ICT) is used to develop customizations for QAD Enterprise Applications. If ICT is included in the environment, customizations developed with ICT are staged in the location configured by the *customizations.ict.dir* setting with the default:

```
customizations/ict
```

Within the customization directory a compile list *utcompil.wrk* enumerates the files that should be compiled with ICT. It may be necessary to add a customized program to this list when using certain ICT features like "program hooks":

```
customizations/ict/utcompil.wrk
```

When a task is closed in a development environment, the task is promoted to the directory configured by the *ict.central.dir* setting, with the default:

```
dist/ict-customizations
```

ICT can be (re)installed into an environment following these steps:

Install and Configure the Integrated Customization Toolkit

Install and Configure the Integrated Customization Toolkit

Prerequisite

The following are required before the Integrated Customization Toolkit can be installed.

- ICT release media (integrated-customization-toolkit + yab-ict)
- ICT licenses

Installation

To install the Integrated Customization Toolkit:

```
> yab install /dr01/installs/ict/integrated-customization-toolkit-4.1.20.0.zip
```

If the environment will be used to develop ICT customizations *ict.toolkit* should be set to true. This is required to co nfigure the PROPATH to support developing customizations in application sessions.

```
ict.toolkit=true
```

Execute the *update* command to apply the changes:

```
> yab update
```

Configuration

Find the location of the qadict.dmp file

Print the location of the *integrated-customization-toolkit* package and use it to find the location of *qadict.dmp* (choos ing the version that end with "_2").

```
yab config packages.integrated-customization-toolkit.dir
packages.integrated-customization-toolkit.dir=/dr01/qadapps/qea/build/catalog/packag
es/integrated-customization-toolkit/4/1/20/0

> find
/dr01/qadapps/qea/build/catalog/packages/integrated-customization-toolkit/4/1/20/0
-name "qadict.dmp"
/dr01/qadapps/qea/build/catalog/packages/integrated-customization-toolkit/4/1/20/0/4
.1.20_1/data/qadict.dmp
/dr01/qadapps/qea/build/catalog/packages/integrated-customization-toolkit/4/1/20/0/4
.1.20_2/data/qadict.dmp
```

Start a character client session

```
> scripts/client-us.sh
```

Load the qadict.dmp file

After logging in, go into the Progress editor by running the function *mgeditor.* In the editor type the following 4GL command and hit F1 to execute the command:

```
run us/ic/icdatmt.p
```

Use the *Load Procedure for Customisation* program that is started to load the *qadict.dmp* file located earlier.

Register the toolkit license (development environments only)

Run License Maintenance (36.16.10.1) to enter your ICT toolkit license.

> You must start a new client session for the license changes to take effect.

Apply control settings

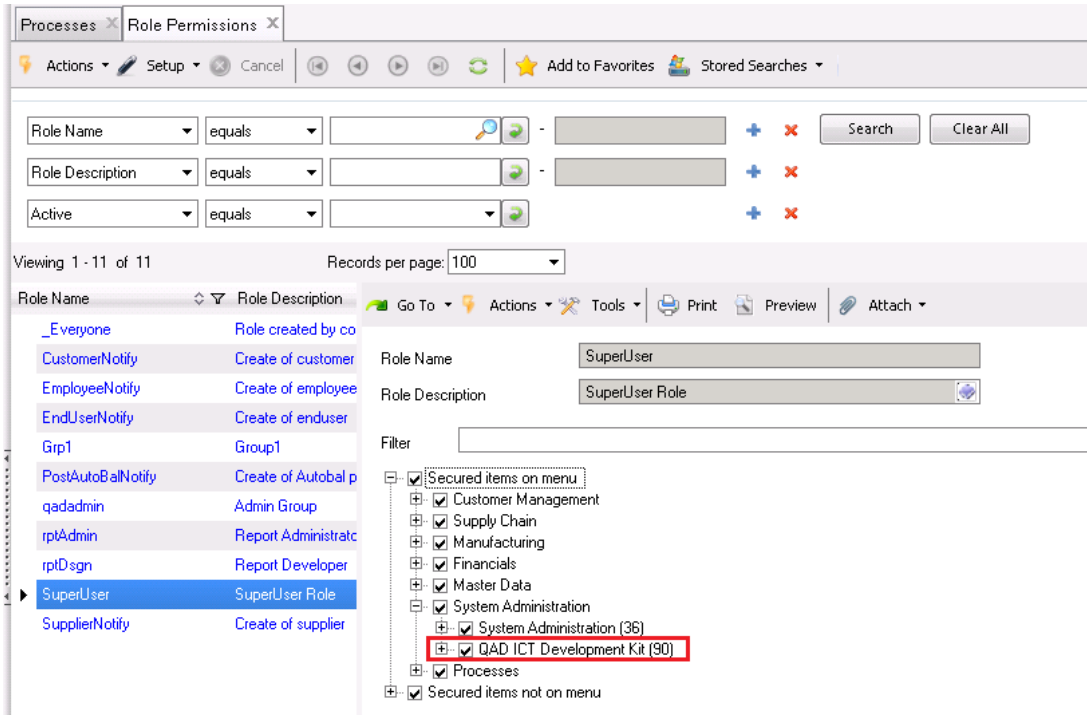
```
> yab -clean module-ict-control-update
```

Process menu changes

```
> yab -topic:Topic15 fin-sync-run
```

Adjust permissions to ICT menus in a .NET UI session

Run Role Permissions Maintain and update permissions to ICT menus.



Desktop Customizations

Desktop customizations typically take the form of Progress programs to support a browse or a report. Customizations are organized in the directory configured by the *customizations.pro.dir* setting, with the default:

```
customizations/pro
```

The files in the customization directory will deployed into the directory configured by the *netui.pro.dir* setting, with the default:

```
dist/pro
```

> You may need to create any nested directories if that is required by your customization.
>
> Ex.
>
> mkdir -p customizations/pro/com/qad/shell/report/reports

The *netui-pro-update* command (or *update*) copies all customizations into dist/pro and then (re)compiles the programs.

```
> yab netui-pro-update
```

Adding Custom Application Server

Additional application servers can be configured. For example, an application server can be dedicated to running MRP to improve the performance of MRP runs. The following configuration defines a new application server that inherits settings from the operational application server.

build/config/configuration.properties

```
# @extends appserver.mfg
appserver.mrp=
appserver.mrp.name=as-mrp
appserver.mrp.operatingmode=State-reset
appserver.mrp.initialsrvrinstance=12
appserver.mrp.maxsrvrinstance=20
appserver.mrp.minsrvrinstance=12
# @port
appserver.mrp.portnumber=23601
appserver.mrp.manual=true
```

To apply or reapply the application server configuration to the system:

```
> yab reconfigure
```

After applying the configuration to the system, the application server could be started and stopped as it is needed.

Start

```
> yab appserver-mrp-start
```

Stop

```
> yab appserver-mrp-stop
```

Status

```
> yab appserver-mrp-status
```

To have the application server start and stop with the rest of the components in the environment adjust the following setting:

```
appserver.mrp.manual=false
```

To display help for all application server settings:

```
> yab help appserver.
```

Adding Custom Data

Use the *data* configuration type to define data to load into the system.

Ex. Loading data in Progress native (.d) format.

```
data.custom.database=db.qaddb
data.custom.dir=/dr01/custom/data
data.custom.format=dotd
```

In the preceding example this configuration would create the command *data-custom-update* to load the data and the command would be processed during a create/update and whenever destination database was rebuilt. The name "custom" was used to identify this set of data but could be any unique value.

To see more options for the *data* configuration type.

```
> yab help data.
```

To list the ID's of databases.

```
> yab config db.* | grep physicalname
```

The *data* configuration type can also be used to load data serialized in QAD XML format.

Patches

Operational Hotfixes (Patches/ECO)

An operational patch can include Progress programs and data. Patches are organized in the directory configured by the *patches.mfg.dir* setting, with the default:

```
patches/mfg
```

Setup

To define a new patch create a sub-directory for the patch:

```
mkdir patches/mfg/patch1
```

Create directories for the source code (as necessary):

```
mkdir -p patches/mfg/patch1/src/us/so
mkdir -p patches/mfg/patch1/src/us/wo
```

Create directories for the data (as necessary):

```
mkdir -p patches/mfg/patch1/data/qadadm
mkdir -p patches/mfg/patch1/data/qaddb
```

Copy the source code into the *src* directory and the data (Progress .d or QAD XML) into the *data* directory.

> As a convenience a *default* patch is pre-defined:
>
> ```
> > tree patches/mfg/default
> patches/mfg/default/
> |-- data
> | |-- qadadm
> | `-- qaddb
> `-- src
> ```

Installing

To install the patch execute the *update* command with the refresh option (-r).

```
> yab -r update
```

The data will be loaded into the relevant databases and the operational code will be recompiled, including the patched sources, into a directory configured by the *code.mfg.dir* setting, with the default:

```
dist/mfg
```

> YAB will automatically order the compile PROPATH so that patches take precedence over standard programs. If the relative order of patches is important a *grouporder* annotation may be defined on the *code.mfg.propath* setting to fine-tune the operational code compile PROPATH. Patches that are not enumerated in the *grouporder* annotation will be ordered after the enumerated patches.
>
> Ex.
>
> ```
> # @grouporder 3 patch2 patch1
> code.mfg.propath=
> ```

Removing

To remove patched programs from the system (source code and rcode):

```
rm -rf patches/mfg/[NAME]/*
yab code-mfg-update
```

Financials Hotfix

Financials is 'hotfixed' by copying the hotfix **r-code** into the following directories:

Proxy hotfix code:

```
dist/fin/proxypatch
```

Financials hot fixes:

```
dist/fin/patch
```

These directories are included in the run time propath property fields.

Financials can also be hotfixed by copying the proxy hotfix **source code** into the following directory:

```
patches/fin/proxypatch
```

This directory is included in the compile time propath property fields.

Hotfixes and Upgrades

Financial patches are not automatically removed during an upgrade so they should be reviewed and updated manually when running an upgrade or a hotfix.

Removing

To remove patched proxy programs from the system (source code and rcode):

```
> rm -rf patches/fin/proxypatch/*
> yab code-mfg-update
```

To remove Financial hotfix code the files should be deleted from

Proxy hotfix code:

```
dist/fin/proxypatch
```

Financials hot fixes:

```
dist/fin/patch
```

and the environment restarted.

Package Patch

A package is patched by overlaying another "patch" package on top of the package, potentially adding and replacing files. The package is patched in the local software catalog.

Configuration

Patches are configured using the *patches* setting:

`packages.[INSTANCE].patches=[PACKAGE RANGE],[PACKAGE RANGE]...`

Ex. Patch the dde-build-1.1.0.0 package by the application of two patch packages.

```
packages.dde-build=1.1.0.0
packages.dde-build.patches=dde-build-patch-1.1.0.1,dde-build-patch
-1.1.0.3
```

Patches are applied to the destination package in the order in which they are defined. If a configuration change invalidates the patched package (i.e. a patch is removed, the patch order changes), the original package will be restored and patched (as necessary).

> A package with patches configured is automatically sourced from the local software catalog.

Displaying

The *info* command lists the configured packages. A package with patches lists the patches below the package.

```
> yab info
...
dde-build                                          1.1.0.0        local
+ dde-build-patch                                  1.1.0.1        remote
+ dde-build-patch                                  1.1.0.3        remote
...
```

Product Configurations

A product configuration is a set of packages whose compatibility has been validated. The product configuration is defined by the system documents:

`build/config/system`

A system can be upgraded by updating the product configuration (pulling in new packages) or by choosing packages directly in the instance document:

`build/config/configuration.properties`

Upgrading the product configuration has the following advantages:

- Ensures the system is using a combination of packages that has been widely used and tested together.
- Avoids typing errors.
- More clearly shows how the system differs from a known base configuration.
- A Product Configuration is the listing that defines the packages to be put into a *bundle* that can be installed by YAB.

Products are upgraded in an environment in a two step process:

1. Configure the new package versions.
2. Execute the update command with the *check-for-updates* option to process the changes.

Ex. Apply package changes (see check-for-updates setting).

```
> yab -check-for-updates update
```

Upgrading Product Configuration

Backup Configuration Files

The following command creates a ZIP file in the working directory that includes the configuration files.

```
> yab system-diagnostics
```

Convert System Configuration Files

A conversion that concatenates all system configuration files into a single file (build/config/system/base).

```
> yab -consolidate-settings:base
```

Review Instance Configuration File

Packages that are configured in the instance document may conflict with packages configured in the new product configuration. Review the instance document for obsolete patches and updates and remove the settings as appropriate.

`build/config/configuration.properties`

The *system-check-for-updates* command can show the impact settings in the instance configuration document would have on the environment were it upgraded to the new product configuration, without having any side-effects on the environment.

```
> yab -local -p1:[NEW PRODUCT CONFIGURATION FILE|URL]@base system-check-for-updates
```

Upgrade

Installs the new product configuration and then upgrades the system.

```
> yab -p:[NEW PRODUCT CONFIGURATION FILE|URL]@base update
```

Upgrading Individual Packages

Configuring

Edit the instance configuration document adding or updating packages settings:

`build/config/configuration.properties`

Ex.

```
packages.fin-src-proxy=2015.0.80.11
```

Applying

Executing the *update* command applies the change.

```
> yab update
```

Troubleshooting

Logging

Log information is recorded to the build/logs directory by default.

YAB

The *yab.log* file records log messages for requests submitted through YAB. The types of messages that will be recorded in the log file can be controlled with a request level option:

`-log-level:[OFF|FATAL|ERROR|INFO|DEBUG|TRACE]`

The level is interpreted as a threshold. A setting of OFF will discard all log messages, whereas the default setting of DEBUG will record FATAL, ERROR, INFO, and DEBUG messages to the log file.

```
> yab -log-level:trace ...
```

> The (-v) option will write log messages to the console as well as to the log.

Changing Default Logging Configuration

The logging in YAB uses the log4j framework. The log4j config file *build/config/etc/log4j.xml* is a standard log4j configuration document that is used to define specific thresholds for logging categories and to determine where log messages are routed (appenders).

Progress

The setting *progress.service.loglevel* is used to set the initial log level of all Progress servers.

Log Levels:

0 - No log file written
1 - Error only
2 - Basic
3 - Verbose
4 - Extended

> The following query shows the current log level of all the Progress servers.
>
> `yab config "*log*level"`

To apply a logging level change run the update command that corresponds to the server.

Ex.

```
> yab appserver-mfg-update
```

Exceptions

The database log files (*.lg) are written in the same directory as the other database files.

Temporary Files

YAB creates and removes temporary files while servicing requests.

Keep Temp Files

To prevent YAB temporary files from getting deleted set the environment variable KEEP_TEMP_FILES to true:

```
export KEEP_TEMP_FILES=true
```

To resume deleting temporary files, unset the variable.

```
unset KEEP_TEMP_FILES
```

Configure Temp Directory

By default temporary files are written into the */tmp* directory. YAB can be configured to write temporary files into a different directory by configuring the YAB_JAVA_OPTS environment variable:

```
export YAB_JAVA_OPTS=-Djava.io.tmpdir=/qond/tmp
```

Collecting Diagnostic Information

To help in diagnosing a problem you may be asked to create a diagnostics archive. The command *system-diagnostics* creates an archive that contains log files and configuration files from the environment. The archive is created in the current working directory using the following naming pattern:

`[APPDIR DIRECTORY NAME]-[TIMESTAMP].zip`

```
> yab system-diagnostics


                        system-diagnostics (1 task)
------------------------------------------------------------------------------
1/1 system-diagnostics                                      OK (6.211 s)
------------------------------------------------------------------------------
BUILD SUCCESSFUL (7.642 s)

> ll *.zip
-rw-rw-r-- 1 mfg qad 1316481 Mar 26  2015 qea-20160311095017.zip


> unzip -l complete-20150326112031.zip
Archive:  complete-20150326112031.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
   310430  03-11-16 09:50   info
      704  03-11-16 09:50   build/logs/yab.log.config
  4952891  03-11-16 09:50   build/logs/yab.log
        0  03-11-16 09:50   build/logs/sql.log
     7590  03-11-16 09:50   build/logs/grs93_wsa.log
    62036  03-11-16 09:50   build/logs/desktop.log
        0  03-11-16 09:50   build/logs/desktop-authentication.log
   457737  03-11-16 09:50   build/logs/admserv.log
   180805  03-11-16 09:50   build/logs/cmdplugin.log
    53541  03-11-16 09:50   build/logs/ns-default.log
    23313  03-11-16 09:50   build/logs/as-crm.broker.log
    28404  03-11-16 09:50   build/logs/as-crm.server.log
    26495  03-11-16 09:50   build/logs/as-eam.broker.log
   543861  03-11-16 09:50   build/logs/as-eam.server.log
    93006  03-11-16 09:50   build/logs/as-fin.broker.log
   893259  03-11-16 09:50   build/logs/as-fin.server.log
...
```

Interactive Execution

The request option (-debug-pause) puts YAB into a mode where the end user is prompted before each step in a command is executed. When prompted the user has the choice of skipping the step, executing the step, executing the step and all subsequent steps, or exiting the request. This mode can be useful when it is necessary to halt or pause the execution at a critical point to examine the state of the system before continuing.

```
> yab -debug-pause update
                            update (313 tasks)
------------------------------------------------------------------------------
1/313   qxtend-stage
Press ENTER to run 'qxtend-stage' ('s' skip, 'a' run all, 'q' quit).
>
SKIPPED (0.003 s)
2/313   adminserver-script
Press ENTER to run 'adminserver-script' ('s' skip, 'a' run all, 'q' quit).
>
OK (0.037 s)
3/313   nameserver-script
Press ENTER to run 'nameserver-script' ('s' skip, 'a' run all, 'q' quit).
> q
------------------------------------------------------------------------------
BUILD CANCELLED (23.590 s)
```

A "breakpoint step" may be specified at which to begin prompting. Steps that precede the breakpoint will be executed without prompting.

```
yab -debug-pause -break:netui-pro-compile update
```

Alternatively the steps that precede the breakpoint may be skipped. This latter approach can be useful to continue a command that consists of many steps at the point of failure.

```
yab -debug-pause -break:netui-pro-compile -skip update
```

Use the system-process-list command to list the "steps" associated with a command:

```
yab system-process-list update
1. qad-rd-permissions-mkdirs
2. adminserver-script
3. nameserver-script
...
```

Skipping Commands

The process-ignore file is used to enumerate commands that should not be executed, for example, because the command is not appropriate for the environment or because the command is not functioning correctly and is preventing the execution of other essential commands.

`build/config/etc/process-ignore`

```
# This file enumerates commands that should not be executed in
this environment.
#
# Each command should be listed on a new line and lines beginning
with
# the '#' character are handled as comments. When asked to run a
# command on the list YAB will skip the command.
#
# Adding a command to the list only ignores that command and has
# no effect on other commands implied by the command. You can use
the
# system-process-list command to enumerate the commands that are
# implied by a command.
#
# Ex.
# yab -nonumber system-process-list start
#
```

The process-ignore file is re-processed when the system is refreshed.

```
> yab -r ...
```

Refresh & Clean Options

YAB uses pre-calculated data to efficiently service requests. Under normal conditions, it correctly determines when the pre-calculated data can be used and when it cannot. The refresh and clean options are used to force YAB to discard or ignore certain pre-calculated data when servicing a request.

Refresh Option

The refresh option (-r) will force YAB to re-evaluate the application configuration potentially resulting in changes to configuration settings, commands, and even the packages associated with the environment. This re-evaluation is automatically triggered whenever a configuration document is added, removed, or modified, the *update* command is executed, or the clean option (see below) is defined.

```
> yab -r ...
```

Clean Option

The clean option (-clean) option discards/ignores the information used by the requested commands to determine if any work needs to be performed.

Ex. Force an update to the assistance help documentation.

```
> yab -clean help-assistance-help-ee-erp-en-update
```

Failonerror Option

The *failonerror* option is used to control whether an error raised while processing a command should halt further processing of the command.

This option is typically used as a request parameter to get around a transient problem. For example, in an environment where one of the servers was no longer controllable, the option might be added to the *stop* command to continue to shut down the environment in a normal manner.

```
> yab -failonerror:false stop

                              stop (16 tasks)
--------------------------------------------------------------------------------
1/16  tomcat-default-stop                                 ERROR   (2.587 s)
2/16  appserver-fin-stop                                  STOPPED (3.930 s)
3/16  appserver-mfg-stop                                  STOPPED (0.993 s)
4/16  appserver-qra-stop                                  STOPPED (0.566 s)
5/16  appserver-qxosi-stop                                STOPPED (0.232 s)
6/16  appserver-qxoui-stop                                STOPPED (0.679 s)
7/16  appserver-qxtnative-stop                            STOPPED (0.328 s)
8/16  webspeed-default-stop                               STOPPED (0.368 s)
9/16  database-qadadm-stop                                STOPPED (0.421 s)
10/16 database-qaddb-stop                                 STOPPED (0.031 s)
11/16 database-qadhlp-stop                                STOPPED (0.027 s)
12/16 database-qadmodule-stop                             STOPPED (0.044 s)
13/16 database-qxevents-stop                              STOPPED (0.022 s)
14/16 database-qxodb-stop                                 STOPPED (0.025 s)
15/16 nameserver-stop                                     STOPPED (0.000 s)
16/16 adminserver-stop                                    STOPPED (0.984 s)
--------------------------------------------------------------------------------
```

Validation Error Setting

The *validation.error* setting controls whether validation errors should halt further processing of a command.

```
validation.error=[TRUE|FALSE]
```

The default setting is true to halt the processing of a request whenever a validation error is detected.

> The *validation.error* setting must be defined in configuration.properties (see Configuration Change Procedure) and cannot be set as a request parameter.

Define Java Startup Parameters

Java startup parameters for the YAB process are defined using the environment variable YAB_JAVA_OPTS.

Ex. Setup remote monitoring of the YAB process

```
export YAB_JAVA_OPTS="-Dcom.sun.management.jmxremote.port=3334
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false"
```

Ex. Define the MaxPermSize setting

```
export YAB_JAVA_OPTS="-XX:MaxPermSize=128m"
```

Appendix

Parallel Execution

YAB commands can be executed in *serial* or in *parallel*. When commands are executed serially, they are executed one at a time, and when they are executed in parallel, more than one command may be executed at a time. By default all commands are configured for serial execution.

> Commands are executed in an order that satisfies all ordering constraints, whether executed serially or in parallel. For example, the start command is executed in parallel, a database may be started without waiting for other databases to be started, but no application server will be started until all databases are started, because of a constraint that ensures that application servers are always started after database servers.

Configuration

Configuring a command to execute in parallel is a two step process:

Configure the command to execute in parallel

Use the following idiom, replacing NAME with the name of the command that should be executed in parallel.

```
process.NAME.id=NAME
process.NAME.parallel=true
```

Ex. Configure the start and stop commands to be executed in parallel.

```
process.start.id=start
process.start.parallel=true
process.stop.id=stop
process.stop.parallel=true
```

Configuring a command to execute in parallel also configures the sub-commands to execute in parallel when executed through the parent command. For example, the command *start* includes the command *database-start*. If *start* were configured to execute in parallel, executing *start* (yab start) would also execute *database-start* in parallel. If *database-start* was executed directly (yab database-start) it would be processed serially.

A command configured for parallel execution can be included within a set of commands configured for serial execution. If *start* were configured to execute in parallel, when *start* was executed by executing update, the commands associated with *start* would be executed in parallel and the commands preceding and following the *start* would be executed serially.

Configure the number of threads to service parallel commands

The *threads* property specifies the number of threads of execution that should be allocated to executing commands that are configured for parallel execution. The default value is 1 which is equivalent to executing all commands serially (whether or not they are configured for parallel execution).

Ex. Allocate up to 4 threads to service parallel commands.

```
threads=4
```

The threads property can be specified as a request parameter.

Ex. Execute the database-backup command allocating 10 threads.

```
> yab -t:10 database-backup
```

The setting *threads.max* is a safety measure to prevent *threads* from being inadvertently set to an inappropriate value. It limits the maximum value accepted for the threads setting and has a default value of 10. This can be increased as needed.

Console Output

When commands are executed in parallel the start and completion of the process are captured on two separate output lines as demonstrated in the following example.

```
> yab -t:2 status
                            status (35 tasks)
------------------------------------------------------------------------------
1/35  adminserver-status                                  PROCESSING
2/35  nameserver-status                                   PROCESSING
1/35  adminserver-status                                  STOPPED (1.348 s)
3/35  database-alerts-status                              PROCESSING
4/35  database-bisgen-status                              PROCESSING
3/35  database-alerts-status                              STOPPED (0.224 s)
4/35  database-bisgen-status                              STOPPED (0.035 s)
2/35  nameserver-status                                   STOPPED (0.358 s)
...
```

Logging

The messages in the log file identify the thread that produced the message:

build/logs/yab.log

```
2015-04-30 06:56:47,460 DEBUG [Thread-9] STDOUT - 06:56:47 BROKER
0: At end of Physical redo, transaction table size is 256. (13547)
2015-04-30 06:56:47,636 DEBUG [Thread-2] BuildContext -
database-qaddb-start STARTED
```

Schema Changes

The following commands update the schema of databases:

Command	Description
`update`	Updates the environment.
`database-update`	Updates all databases.
`database-INSTANCE-update`	Updates a specific database.
`database-INSTANCE-schema-update`	Updates the schema of a specific database.

Each database is associated with zero or more files defining the schema for the database. If the -clean option is used, a new file is added, or an existing file changes, the database will be reevaluated. Reevaluation consists of comparing the configured schema with that of the target database producing an incremental schema file that is capable of upgrading the target database to the configured schema:

`[DATABASE DIRECTORY]/.yab/[PHYSICAL NAME]/schema/incremental.df`

> When an environment is created or a database is rebuilt, the schema files are directly applied to the database without first generating an incremental schema file as an optimization.

> The default behavior is to compare the configured schema files to the target database. If tables or sequences are defined in the database outside of the system they will be removed when the database is reevaluated. The following setting can change this behavior for a database at the expense of additional processing time.
>
> `db.INSTANCE.allowunrecognizedschema=true`
>
> NOTE: This only applies to tables and sequences. Fields, triggers, and indexes added to tables recognized by the system will always be removed if the system is unaware of the additions. Reference the *incremental* property on the *schema* type (below) for more information on configuring the system to be aware of these changes.

To list the schema configuration:

```
> yab config schema.*
schema.fin-qaddb-qadcpl.database=db.qadcpl
schema.fin-qaddb-qadcpl.file=/dr01/qadapps/qea/build/catalog/packages/fin-ee2016/201
6/0/16/209/schema/qadfin/finempty.df
schema.fin-qaddb-qadcpl.version=2016.0.16.209
...
```

To display the help for schema settings:

```
> yab help schema.

SETTINGS

     schema.area.map                Used to map the AREA(s) defined in the schema
to the AREA(s) defined
                                    in the target database.

                                    Mapping cannot be performed on incremental
schema files.

     schema.database                A reference to the database where the schema
should be applied.

     schema.dir                     The root directory containing the schema files
to process.

                                    Use either 'file' or 'dir' to locate schema
files.

     schema.direct                  Deprecated

                                    This is a synonym for incremental.

     schema.excludes                A comma-delimited list of exclude patterns to
match schema files
                                    in the configured directory.

     schema.file                    The schema file to process.

                                    Use either 'file' or 'dir' to locate schema
files.

     schema.includes                A comma-delimited list of include patterns to
match schema files
                                    in the configured directory.
                                    Default: **/*.df

     schema.incremental             Whether the schema files are incremental or
full.

                                    Full schema files should be configured allowing
the system to calculate the
                                    incremental difference between the schema file
and the target database. The
                                    only exception to this rule is when adding
fields or triggers to tables
                                    defined in another schema file.

     schema.objects.excludefiles    A comma-delimited list of files listing schema
objects to exclude, one per line.

                                    Filtering cannot be performed on incremental
schema files.

     schema.objects.excludes        A comma-delimited list of schema objects to
exclude.

                                    Filtering cannot be performed on incremental
schema files.

     schema.objects.includefiles    A comma-delimited list of files listing schema
objects to include, one per line.

                                    Filtering cannot be performed on incremental
schema files.

     schema.objects.includes        A comma-delimited list of schema objects to
include.

                                    Filtering cannot be performed on incremental
schema files.
```

Configuration Files

Files

Administrative settings are defined in configuration files that are processed to create the application configuration.

Type	Location	Role
Instance	build/config/configuration.properties	Instance specific settings.
System	build/config/system	Base product configuration + environment settings.
Package	build/config/packages	Factory default settings distributed in packages.

The instance document settings override the system document settings which in turn override the package document settings. The config.order setting defines the precedence order between system documents. Configuration files can reference settings defined in other configuration files.

Format

Settings are defined one per line using the syntax:

`KEY=VALUE`

Ex.

```
ns.name=ns-default
ns.allowruntimeupdates=0
ns.autostart=1
ns.brokerkeepalivetimeout=30
ns.classmain=com.progress.nameserver.NameServer
ns.collectstatsdata=0
ns.environment=${name}
ns.hostname=${host}
...
```

Keys

The convention is to use names with lowercase characters and the period as a delimiter for compound names (e.g. *dbserver.module.afterimagebuffers*).

Comments

A comment line is started with the hash (#) character.

```
# The physical name of the qaddb database.
db.qaddb.physicalname=qadprod
```

References

Values may reference other settings.

`${[REFERENCE]}`

Ex.

```
mongodb.default.logfile=${appdir.logs}/${name}.mongodb.log
```

Resolution

A reference is resolved using the following algorithm:

- Resolve the reference relative to the parent of the referencing key.
- Resolve the reference as an absolute reference.
- Resolve the reference as a Java system property.
- Replace the reference with an empty string.

Ex.

```
foo.bar=Hello ${user}
```

1. Replace "${user}" with the value of the setting *foo.user* if defined.
2. Replace "${user}" with the value of the setting *user* if defined.
3. Replace "${user}" with the value of the Java system property *user* if defined.
4. Remove "${user}".

Nesting

References can be nested.

Ex. russian.dolls=d

```
a=d
b=a
c=b
russian.dolls=${${${c}}}
```

Escaping

To prevent YAB from interpreting a literal "${" as a property reference, the "$" can be escaped by doubling the "$" character.

Ex.

```
foo.bar=Hello $${user}
```

Dereferencing Aliases

The following syntax can be used to dereference settings that might be aliased (see below @aliasof)

`${<[REFERENCE]}`

Ex. d=c and e=b

```
a=>c
b.color=orange
c.color=blue
d=${<a}
e=${<b}
```

Annotations

Annotations define additional information about a setting or a group of settings using the syntax:

```
# @[ANNOTATION NAME]
# @[ANNOTATION NAME] [ANNOTATION VALUES...]
```

@if

Asserts a test to be evaluated.

If the test evaluates to FALSE and the test IS final, the configuration document will not be processed any further. If the test evaluates to FALSE and the test IS NOT final, the properties that follow the test will be skipped until an @end annotation is defined or the end of the file is reached.

NOTE: Tests that reference configuration settings should only reference settings defined by higher precedence sources, because tests are evaluated immediately as sources are processed.

Variants

```
# @if [TEST]
# @if FINAL [TEST]
```

@unless

Asserts a test to be evaluated.

If the test evaluates to TRUE and the test IS final, the configuration document will not be processed any further. If the test evaluates to TRUE and the test IS NOT final, the properties that follow the test will be skipped until an @end annotation is defined or the end of the file is reached.

NOTE: Tests that reference configuration settings should only reference settings defined by higher precedence sources, because tests are evaluated immediately as sources are processed.

Variants

```
# @unless [TEST]
# @unless FINAL [TEST]
```

@end

Ends the scope of one or more tests.

If/Unless Tests

Test	Example	Description

KEY	foo	Tests if the property 'foo' is defined.
KEY = VALUE	foo = bar	Tests if the value assigned to the property 'foo' equals 'bar'.
KEY ~ VALUE	foo ~ bar	Tests if the value assigned to the property 'foo' contains 'bar'.
KEY < VALUE	foo < bar	Tests if the value assigned to the property 'foo' is less than 'bar'. (Uses a numeric comparison if appropriate.)
KEY <= VALUE	foo <= bar	Tests if the value assigned to the property 'foo' is less than or equal to 'bar'. (Uses a numeric comparison if appropriate.)
KEY > VALUE	foo > bar	Tests if the value assigned to the property 'foo' is greater than 'bar'. (Uses a numeric comparison if appropriate.)
KEY >= VALUE	foo >= bar	Tests if the value assigned to the property 'foo' is greater than 'bar'. (Uses a numeric comparison if appropriate.)
KEY =~/REGEX/	foo =~/bar/	Tests if the value assigned to the property 'foo' is completely matched by the regex '/bar/'.
KEY =* VERSION RANGE	foo =* [1.2)	Tests if the value assigned to the property 'foo' is a version that is a member of the '[1.2)' version range.

The built in tests can be extended by referencing an implementation of the IConditionEvaluator interface using the following syntax:

Test	Example	Description
<CLASS> TEST	<com.qad.yab.FibonacciNumberEvaluator> 4	Delegates the evaluation of the test '4' to an instance of the 'com.qad.yab.FibonacciNumberEvaluator' class.

@extends

Used to "inherit" the child keys of another key.

Variants

The key to extend.

```
# @extends [KEY]
```

In the following example, bar would inherit the property (bar.greeting=Hello) from foo while preserving (bar.farewell=Hasta Luego).

```
foo.greeting=Hello
foo.farewell=Goodbye

# @extends foo
bar=
bar.farewell=Hasta Luego
```

@aliasof

Used to make a key an alias of another key.

Defining a key as an alias of another key places the two keys in an equivalence relationship where they have identical child properties. If we use A to represent the key defining the alias and B to represent the key being aliased, then this equivalence relationship is formed by adding to A all child properties of B which are not already defined in A and adding/replacing all child properties in A with those defined in B. Subsequent updates to the configuration may violate this relationship.

Some operations to enumerate configuration instances will not include aliases in the enumeration.

Variants

The key to alias.

```
# @aliasof [KEY]
```

Alternatively aliases can be expressed without annotations using the following notation:

or

```
[ALIAS KEY]=>[ALIAS OF KEY]
```

In the following example, bar would inherit the property (bar.greeting=Hello and bar.farewell=Goodbye) from foo while foo would inherit the property (foo.baz=true) from bar.

```
foo.greeting=Hello
foo.farewell=Goodbye

# @aliasof foo
bar=
bar.farewell=Hasta
bar.baz=true
```

To escape a configuration value that begins with the '>' character use the '\':

```
foo2=\>foo
```

@ref

Used to add the configuration of another key to this key.

A key may reference multiple keys. A reference may be used to inherit appended values (see @append) while preserving any ordering information (see @group). In the following example, a would be set to (a=foo,bar,baz,end).

```
# @append
# @group 3
d=end

# @append
# @group 1
c=bar

# @append
# @group 2
# @ref c
b=baz

# @append
# @group 0
# @ref b
# @ref d
a=foo
```

Variants

The key to reference.

```
# @ref [KEY]
```

@append

Used to indicate that the value should be appended to the existing value if the property is already defined (normally it is discarded).

If multiple sources define an append delimiter for the property, the delimiter defined by the source with the highest precedence will be used, otherwise if no sources define the delimiter, a comma will be used.

Java character escape codes can be used to represent whitespace delimiters:

Escape	Character
\n	Newline
\t	Tab
\u0032	Space

Variants

The value should be appended to an existing value using a comma as the delimiter.

```
# @append
```

The value should be appended to an existing value using the specified delimiter.

```
# @append [DELIMITER]
```

@noappend

Used to prevent values from being appended to the property.

The annotation only has an effect when it is defined by the highest precedence source for the property.

Variants

Any appended values should be discarded.

```
# @noappend
code.mfg.databases=db.qadcpl,db.qadadm,db.qadhlp,db.qadrcode
```

@group

Used in conjunction with @append to obtain a desired value ordering when the order matters.

The appended value is guaranteed to be preceded by all values associated with lower group indexes and followed by all values associated with a higher group indexes. Values associated with the same group are ordered by source precedence unless ordering hints or the *@grouporder* annotation is defined.

An ordering hint is defined using the syntax *[+|-]ID* where the *ID* corresponds to the *@id* of another group member and (-) is interpreted as an instruction to order the appended value before the referenced value and (+) as an instruction to order the appended value after the referenced value. When the *@grouporder* annotation is defined for a group it takes precedence over all ordering hints for that group.

The meaning of a group index depends on the nature of what is being configured. We have given the following interpretation to group indexes in the context of configuring a PROPATH.

0=Configuration
1=Bootstrap
2=Customizations
3=Patches
4=Standard *****

***** The standard group is the implied group when a group is not defined. To allow for the possibility of introducing additional groups in the future, members of the standard group should not explicitly define a group identifier.

Variants

Assign the value to the lowest precedence group.

```
# @group
```

Assign the value to the group.

```
# @group [INDEX]
```

Assign the value to the group and define some ordering hints.

```
# @group [INDEX] [ORDERING HINT] [ORDERING HINT]...
```

@grouporder

Used to assign a secondary ordering to the members of a *@group*.

The group members will be ordered in the listed order. Members not specified in the list will be ordered after the listed members by source precedence. IDs that do not correspond to a member of the group will be ignored.

Variants

Orders the members of the default (lowest precedence) group:

```
# @grouporder [ID] [ID]...
```

Orders the members of a specific group:

```
# @grouporder [INTEGER] [ID] [ID]...
```

@port

Identifies the property as a TCP-IP port.

Variants

Identifies the property as a port.

```
# @port
```

Identifies the property as a port and configures a preferred range offset.

```
# @port +[OFFSET]
```

Identifies the property as a port and configures a default port.

```
# @port [DEFAULT PORT]
```

Identifies the property as a port and configures a preferred range offset and default port.

```
# @port +[OFFSET] [DEFAULT PORT]
```

Assigns a specific port number to a property, using the annotation so YAB will not assign the port to another property.

```
# @port
some.port=30000
```

Assigns a well known offset to ports that users might interact with. If for some reason, the port is already allocated the property will be allocated an available port from the configured port range.

```
# @port +0
tomcat.default.httpport=
# @port +1
adminserver.port=
```

Ports which are not typically relevant for users (the majority of ports) should be configured without an offset. To minimize unnecessary conflict with ports assigned by offset, ports that will accept any valid port are allocated from the upper bound of a port range counting backwards.

```
# @port
tomcat.default.serverport=
# @port
adminserver.adminport=
```

A default port may be defined to handle cases where a port range is not configured.

```
# @port +0 22000
tomcat.default.httpport=
```

@exclude

Used to exclude/ignore the setting and all child settings.

Ex. Excludes the setting 'foo.bar' and 'foo.bar.baz' but not 'foo.barbaz'.

```
# @exclude
foo.bar=
```

@patch

Used to "patch" a property value using a regular expression to identify the character sequences to replace.

When the property value is evaluated all character sequences matched by the regular expression are replaced with the patch value. If the patch value is defined and the regular expression did not match the property value, the patch value will be appended to the property value using the default delimiter ',' or the delimiter specified. If the patch value is not defined, all character sequences matched by the regular expression will be removed from the property value.

Variants

The key to alias.

```
# @patch /[REGEX]/
# @patch /[REGEX]/[REGEX FLAGS]
# @patch /[REGEX]/ [DELIMITER]
# @patch /[REGEX]/[REGEX FLAGS] [DELIMITER]
```

Java character escape codes can be used to represent whitespace delimiters:

Escape	Character
\n	Newline
\t	Tab
\u0032	Space

Ex.

```
progress.startup.params=-T ${tmp} -yy ${progress.centuryoffset} -s 32768 -mmax 8192
-inp 32000 -rereadnolock -c 30 -D 1000 -Bt 350 -nb 200 -cpcoll ${db._base.collation}
-cpcase basic -h 25 -tok 4096 -tmpbsize 8 -TB 31 -TM 32
```

```
# @patch /-Bt [0-9]+/ \\u0032
progress.startup.params=-Bt 3000
```

Configuration File Includes

Configuration files that are brought into the environment using the (-p) option can define include statements that reference other configuration documents.

Ex.

```
yab -p:[URL|FILE] create
```

```
yab -p:[URL|FILE] update
```

The include statement will be replaced by the referenced document in the referencing document.

Include Statement Syntax

```
"{" [DIRECTIVE]","["!"DIRECTIVE]... "|" [URL|ABSOLUTE PATH|RELATIVE PATH] "}"
```

Ex.

```
{core.properties}
{../../envs/rd.properties}
{PROD|production.properties}
{DEVEL,PROD|production.properties}
{http://server/yab/ee/2016/core.properties}
```

Where:

DIRECTIVE	Directives are used to define the conditions under which the referenced file should be included. The client defines certain directives (e.g. WIN, UNIX) as determined by the host platform, other directives can be supplied with the request. The client will assume that all request options in upper case are directives. Directives can be negated in an include statement by prefixing the directive with "!". Directives are optional. If no directives are defined, the pipe delimiter can be omitted. Ex. `yab -p:/opt/media/ee2016.properties -DEVEL -PROD -v create`
URL	A well formed URL (e.g. "http://server/yab/ee/2016/core.properties").
ABSOLUTE PATH	An absolute path to a file (e.g. "/dr01/configurations/core.properties").
RELATIVE PATH	A relative path (e.g. "configurations/core.properties"). The path is resolved relative to the including document. The notation "../" is used to signify a parent path.

Query string parameters that were defined when requesting the root document will be used when requesting the included documents.