



QAD Adaptive Applications
YAB 1.10

Configuration and Administration Guide QAD Enterprise Edition

70-3346-YAB1.10

QAD Adaptive Applications

Enterprise Edition

September 2019

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

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

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

Copyright © 2019 by QAD Inc.

QAD Inc.

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

Table of Contents

Administration Guide	7
Introduction	8
YAB Client	9
Packages	11
Commands	13
Configuration	16
File System Layout	18
Enterprise Edition	19
YAB Principles	20
System Administration	22
Starting and Stopping Environments	23
Controlling Financials Daemons	25
Schema Management	28
Data Management	30
Environment Backup	31
Environment Restore	32
Configuration Backups	33
Data (Native Format)	34
Data (XML Format)	35
Compiling Source Code	37
Configuring Compiles	38
Compile for Developers	39
Locating Compile Resources	41
Generating XREF output	43
Generating DEBUG-LIST output	44
Compile Databases	45
Auditing	46
Enable Auditing	47
Disable Auditing	48
Importing Policy Files	49
Archiving Records	50
Manage Archive Database Server	52
QXtend	53
Reapply QXtend Defaults	54
Enterprise Edition Best Practices	55
Trim Application Servers	56
Database Security Management	57
Enabling Database Authentication	58
Enabling Database Table Level Permission	59
Key Commands	60
clean	61
code-mfg-customizations-update	62
code-mfg-update	63
config	64
database-storage-area-resolve	66

env-info	70
fin-sync-run	71
help	72
host-update	77
info	78
netui-pro-update	80
reconfigure	81
restart	82
shell	83
start	85
status	86
stop	87
system-diagnostics	88
system-find	89
system-package-create	90
system-process-list	92
system-updates-list	93
update	100
validate	101
System Configuration	102
General Information	103
Configuration Type System	104
Configuration Files	106
Distributing Settings	117
Configuration File Includes	118
Undefined Progress Parameters	119
Reconfiguring Databases	120
Configuring Database Location	121
Configuring Database Structure	122
Configuring 4GL Broker Network Support	123
Configuring a SQL-92 Secondary Login Broker	124
Configuring Before Imaging	125
Configuring After Imaging	126
Configuring Codepage and Collation	128
Schema Changes	129
Using Demo Data	132
Using Parameter Files	134
Reconfiguring Tomcat	136
Enabling the Manager Web Application	137
Reconfiguring Tomcat Startup Parameters	138
Configuring SSL Connections	139
Reconfiguring PROPATHs	141
Reconfiguring AdminServer	144
Reconfiguring Application Servers	145
Enabling and Disabling 4GLTrace	146
Migrating to a New Host	147
TCP/IP Ports	148
Adding Languages	149
Using AIA	151
SSH and Telnet Protocols	152

Bootstrap Files	153
QAD Reporting Framework Printers	155
Batch Jobs	156
Batch Jobs (Basic)	157
Batch Jobs (Enhanced)	158
Configuring Commands	160
Exec Commands	161
ANT commands	162
Error Handling	163
Timeouts	164
Key Settings	165
catalogs setting	166
check-for-updates setting	167
clean setting	168
config.order setting	169
dependency settings	170
packages setting	172
ports setting	174
verify setting	175
Upgrades, Customizations, and Patches	178
Product Upgrades	179
Customizations	180
Operational Customizations	181
Integrate QAD source file archive into environment.	
.....	183
Financials Customization	185
Integrated Customization Toolkit	186
Install and Configure the Integrated Customization Toolkit	187
Desktop Customizations	189
Adding Custom Data	190
Metadata	191
Process Maps	192
Adding Custom Database	193
Adding Custom Application Server	195
Compiling EDI Functions	196
Adding Custom Directory Backup Processes	197
Patches	200
Operational Hotfixes (Patches/ECO)	201
Financials Hotfix	202
Package Patch	203
File Patch	204
qra.pl	205
Prepared Contexts	206
Using a context	207
Adding files to a context	209
Validating a context	210
Logging in a context	211
Context Guidelines	212
Limitations	214

Troubleshooting	215
Logging	216
Financials Logging	217
Temporary Files	218
Collecting Diagnostic Information	219
Interactive Execution	220
Disabling Commands	222
Refresh & Clean Options	223
Validation Settings	224
Defining Java Startup Parameters	225
Adding Packages to Local Catalog	226
Appendix	227
Parallel Execution	228
Command Expressions	230
Selecting Files	232
Extensions	233
Rebuild	235

Administration Guide

This is the Administration Guide for QAD Enterprise Edition using Your Application Builder (YAB).

YAB is used to administer QAD Enterprise Edition releases starting with *QAD Enterprise Edition 2015 (QAD Cloud)* and is a functional replacement for the *QAD Deployment Toolkit (QDT)*. This guide documents relevant information regarding the configuration and administration of the QAD Enterprise Edition. It is oriented towards system administrators with a basic understanding of the QAD Enterprise Edition.

The *QAD Enterprise Edition Installation Guide* provides instructions for installing the Enterprise Edition and is available in the QAD Document Library:

<http://documentlibrary.qad.com>

Documentation Conventions

To provide the clearest exposition of the subject matter, the examples in the guide are written from the following perspective:

- The `yab` command is available on the PATH.
- The working directory is the installation directory.
- The default file system layout is used.

Introduction

The *Introduction* section introduces the key concepts behind YAB.

YAB Client

A QAD Enterprise Edition environment is administered with the program *yab*.

Executing the program without any arguments (or with the "-?" option) will display the version of the YAB client and its help.

```
> yab
YAB Console, 1.10.0.45

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

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 a configuration document to integrate into 'build/config
/system'. (multiple allowed)

-i:arg          A FILE locating a package to integrate into 'build/catalog'. (multiple allowed)

-r             Forces the evaluation of settings (-r or -r:true) or prevents the evaluation of
settings (-r:false). By default settings will automatically be evaluated as needed.

-clean         Used to force certain updates.

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

                  Default: DEBUG

-log-copy       Records a copy of all log messages in a file, or if a file is not specified, in
a timestamp file in the log directory.

-?            Prints this help (or -?? for additional options).

Arguments:

COMMAND        A command to execute.
```



The version that is displayed (e.g. "YAB Console, 1.6.0.4") is the version of the YAB client, which may be different from the version of YAB in an environment. The [info](#) command displays the version of components in an environment, where the entry `yab-ee-app` records the version of YAB.



When QAD Enterprise Edition is installed or YAB is updated, the installer installs the YAB client into the *build/client* directory. If the YAB client is not on the PATH it will, as a convenience, create a script named `yab.[bat]` in the installation directory to launch the YAB client. It is recommended that you add the YAB client to the PATH so YAB commands can be submitted from any directory.

```
export PATH=[INSTALLATION DIRECTORY]/build/client:$PATH
```

A single installation of the YAB client may be used to administer multiple QAD Enterprise Edition environments, where the environment to administer is selected using the (-a) option. Alternatively the "active" environment can be implicitly determined by navigating to the installation directory of the environment (or a subdirectory of the installation directory).

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

If the client cannot locate the environment to administer, the following error is raised.

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

The YAB client is a Java application that requires Java 8. The *java* program is located using the PATH environment variable, unless the JAVA_HOME environment variable locates an alternative Java installation to use. If Java is not installed or configured correctly you will get an error similar to this:

```
> yab
/users/wtb/programs/yab/yab: line 38: java: command not found
```

If an older version of Java is configured you will get an error similar to this:

```
> yab
Exception in thread "main" java.lang.UnsupportedClassVersionError: Bad version number in .class
file
```

**Show Java Version**

The following command displays the version of Java resolved from the PATH:

```
java -version
```

Packages

An environment is composed of *packages*. Packages are read-only versioned releases of application and management software.

Software Catalogs

A software catalog is a structured collection of packages. A catalog may contain compressed or uncompressed packages. Every QAD Enterprise Applications environment is associated with a local catalog that contains uncompressed packages.

Ex. List contents of local catalog

```
> 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](#) can be used to configure additional catalogs (listed in order of precedence).

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

Downloading Packages

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

Usage

The files in a catalog are referenced by YAB and also by the Enterprise Edition. For example, the default configuration references Tomcat binaries from the *tomcat* package.

Listing Packages

The [info](#) command lists the packages configured in an environment.

```
> yab info

INSTANCE

/dr01/qadapps/ci

MODULES

|- assistance-help-ee                2018.0.0.0      local
|- base-app                         2.4.0.8        local
|- costing-app                      2.4.0.6        local
|- custrelmgmt-app                  1.0.0.83       local
|- demo-data                        2019.0.19.787  local
|- distribution-app                 1.3.0.5        local
|- ee-pre-conversion-reports        1.0.0.2        local
|- engineering-app                  2.4.0.5        local
|- enterprise-financials-app        2019.1.1.2     local
|- fhd-data-fix                     1.0.0.5        local
|- financials-qra-app               2.20191.0.385  local
|- fixed-assets-app                 2.2.0.126     local
|- inventory-app                    2.4.0.13       local
...
```

Creating Packages

The [system-package-create](#) command is used to create a custom package.

See:

[Info](#)

[Clean](#)

[Product Upgrades](#)

[system-package-create](#)

Commands

Commands are executed through the YAB client:

```
> yab [COMMAND] [COMMAND]...
```

Detailed information is recorded in the YAB log file:

```
build/logs/yab.log
```

Use the verbose (-v) option to also print log messages to the console:

```
> yab -v ...
```

Use the (-log-copy) option to write log messages to a file as well as to the YAB log file, replacing existing content:

```
yab -log-copy:request.log ...
```

Many commands are aggregated. For example there are commands to start a single database server, all database servers, and the environment.

Ex. Starts the QADDB database.

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

Ex. Starts all databases.

```
> yab database-start
```

Ex. Starts the environment (including databases).

```
> yab start
```

The product configuration will have an influence on the commands that are available in an environment. For example, when Customer Relationship Management is added to an environment, additional commands are defined.

Ex. Starts the CRM application server

```
> yab appserver-crm-start
```

The [help](#) command is used to show the help for specific commands and to query the available commands.

```

> yab help config

PROCESS
  config - Prints configuration settings.

DESCRIPTION
  yab config [KEY] [KEY] [KEY]...

  KEY
  The key of the setting(s) to print. The meta character '*' is used to match 0 or more
characters.
  If no keys are defined, all configuration settings with a value are printed, and when (-all)
is
  defined, settings without a value are included as well. When using meta characters, quote the
argument to prevent the shell from evaluating the meta character.
...

> 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-deactivate       Perform an index deactivate.
  database-qaddb-index-rebuild         Perform an index rebuild.
...

```



The help command also shows the help for configuration settings. Sometimes this is distracting when commands and settings are named similarly. The command -help command only shows help for commands.

The [system-process-list](#) command lists the commands that are aggregated (implied) by a command and the order in which they would be executed if the parent command was executed.

```
> yab system-process-list database-qaddb-update

1. database-qaddb-structure-file-update
2. database-qaddb-structure-validate
3. database-qaddb-create
4. database-qaddb-structure-update
5. qxtend-stage
6. database-qaddb-schema-update
7. database-qaddb-schema-index-rebuild
8. oid-code-update
9. data-customizations-default-dotd-qaddb-update
10. data-customizations-default-xml-qaddb-update
11. data-ee-data-qaddb-update
12. data-ee-data-qadfin-update
13. data-ee-system-data-qaddb-update
14. data-ee-system-data-patch-delete-qaddb-update
15. data-ee-system-data-patch-qaddb-update
16. data-options-qaddb-update
17. data-patches-default-dotd-qaddb-update
18. data-patches-default-xml-qaddb-update
19. data-qrabridge-qaddb-update
20. data-qxtend-qaddb-update
21. module-archiving-stage
22. data-xml-archiving-qaddb-overwrite-update
23. languages-mark-installed
24. database-qaddb-data-update
25. tlc-create
26. database-qaddb-jta-enable
27. database-qaddb-update
```



Occasionally the term *process* is used instead of *command* to capture the idea that a command describes a process flow. The two terms are interchangeable.

See:

[help](#)

[system-process-list](#)

Configuration

The configuration of the environment is distributed across multiple [configuration files](#) residing in the *build/config* directory. To change a configuration setting the setting should be added to or updated in the *build/config/configuration.properties* file, where it will override the default setting.

For example, to increase the maximum number of databases servers for the QADDB database to 15, the following setting would be added:

```
build/config/configuration.properties

appserver.mfg.maxsrvrinstance=15
```

Certain basic configuration changes can be applied by executing the *reconfigure* command.

```
> yab reconfigure
```

Other configuration changes require the environment to be restarted and are applied with the *update* command.

```
> yab update
```

A setting that is defined at the command line will be used only for the duration of the request.

```
> yab -threads:3 stop
```

Layout

Type	Location	Description				
Instance Document	build/config/configuration.properties	Used to make changes to the standard configuration.				
System Documents	build/config/system	Documents that define a standard configuration of the QAD Enterprise Edition.				
		When the environment is created using the QAD Enterprise Edition installer the following configuration documents are created.				
		<table><tr><td>build/config/product.properties</td><td>The packages that were installed.</td></tr><tr><td>build/config/environment.properties</td><td>The <i>install.conf</i> file used for the installation.</td></tr></table>	build/config/product.properties	The packages that were installed.	build/config/environment.properties	The <i>install.conf</i> file used for the installation.
		build/config/product.properties	The packages that were installed.			
		build/config/environment.properties	The <i>install.conf</i> file used for the installation.			
WARNING: The system will process any file in this directory that ends with ".properties" as a configuration file. If you create a backup of a configuration file, copy the backup into a different directory (or better still use the <i>system-config-backup</i> command to make the backup).						
WARNING: In general, these files should not be edited.						
Package Documents	build/config/packages	Factory default settings defined by packages. WARNING: These files are generated and should never be edited.				

Query

The [config](#) command is used to query configuration settings:

```
> yab config appserver.mfg.*
appserver.mfg.aia=aia.default
appserver.mfg.aiaurl=http://vmwtb02.qad.com:22000/aia/Aia?AppService=as-mfg
appserver.mfg.allowruntimeupdates=1
appserver.mfg.appservicename=as-mfg
appserver.mfg.autostart=0
appserver.mfg.brkrlogginglevel=3
appserver.mfg.brokerlogfile=/qad/sandbox/user/wtb/02/sc2-core/build/logs/as-mfg.broker.log
appserver.mfg.connectionprotocol=appserverdc
appserver.mfg.controllingnameserver=ns-default
appserver.mfg.environment=as-mfg
appserver.mfg.hostname=vmwtb02.qad.com
appserver.mfg.initialsrvinstance=1
appserver.mfg.manual=false
appserver.mfg.maxsrvinstance=5
appserver.mfg.minsrvinstance=1
appserver.mfg.name=as-mfg
appserver.mfg.operatingmode=Stateless
appserver.mfg.portnumber=22098
...
```

The system maintains a single file enumerating the current, fully resolved configuration settings:

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

The [help](#) command shows the help for configuration settings.



The `help` command also shows the help for commands. Sometimes this is distracting when commands and settings are named similarly. The `config-help` command only shows help for configuration settings.

See:

[config](#)

[help](#)

[update](#)

[validate](#)

[Configuration Files](#)

File System Layout

The default configuration creates an environment with resources 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 2 mfg mfg 4096 Mar 17 05:33 extensions/
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/
drwxrwxr-x. 4 mfg mfg 4096 Mar 17 09:22 storage/
```

The location of some resources can be configured but the location should be configured when the environment is created.

Factory Default Path	Configuration Setting	Configurable	Description
	appdir	Yes	The installation directory that was chosen when the application was installed. This is equivalent to the installation variable (install-instance).
build	appdir.build	No	Used by the management components.
build/catalog	appdir.catalog	No	The local software catalog .
build/config	appdir.config	No	The management configuration settings .
build/work	appdir.work	No	A work area for the management components.
build/logs	appdir.logs	Yes	The environment log files.
customizations	customizations.dir	Yes	Application customizations .
databases	db_base.dir	Yes	The databases.
dist	dist.dir	Yes	The compiled application binaries.
extensions	extensions.dir	Yes	Application extensions .
patches	patches.dir	Yes	Application patches .
scripts	scripts.dir	Yes	Scripts to start and stop servers.
servers	n/a	No	Configuration for servers.
storage	qad-qracore.qra.attachment.storage	No	Storage for attachments.

Enterprise Edition

The environment should be started before running an Enterprise Edition client.

```
> yab start
```

Character

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

Start a character session.

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

Ex.

```
> scripts/client-us
```

.NET

The .NET client must be installed on a Windows host. The .NET client installation is started by navigating to the Home Server in a web browser. Use the [config](#) (or [env-info](#)) command to print the Home Server URL:

```
> yab config webapp.homeserver.url webapp.homeserver.secureurl  
webapp.homeserver.url=http://vmwtb02.qad.com:22000/qadhome  
webapp.homeserver.secureurl=https://vmwtb02.qad.com:22077/qadhome
```

YAB Principles

YAB has two main objectives:

- Increase the efficiency of Enterprise Edition administrators.
- Provide a safe and consistent approach to reconfiguring, upgrading, and administering the Enterprise Edition.

These objectives are promoted by certain design decisions (e.g. catalogs and packages) embedded in the fabric of how YAB works, but they are also forwarded by other principles that require the participation of administrators. In many cases, these principles have deep roots and correspond to how Enterprise Edition has been administered in the past. In a few cases, a conscious decision was made to improve upon historical practices.

Principles

Automation

Change should be automated. Automated changes are more efficient than manual changes and produce consistent results that can be reproduced. Automated changes record evidence in log files that can be analyzed for root cause analysis.

Declarative

Configuration should describe the desired state of the environment. It should be possible to reproduce this state at any point in time, which means that all files referenced from the configuration (e.g. schema files, source code, etc) must always be available. The [update](#) command is responsible for ensuring that the environment corresponds to the configuration.

Fault Tolerant

An administrator should be able to start and stop the environment even if YAB is broken. YAB generates scripts to control the environment and uses these scripts internally when starting and stopping the environment.

YAB Best Practices

Use Multiple Environments

A common practice is to have one or more environments dedicated to development and testing which closely resemble the production environment. Changes are validated in advance before they are applied to the production environment.

Avoid Manual Changes

It is very difficult to obtain consistent outcomes when manual (and sometimes undocumented) changes are applied to an environment, for example, manually loading a schema file into a database or compiling a program directly in the Progress procedure editor. There is also the risk that YAB will remove/override the change at some point in the future when it is updating the environment. Consider the case of a manually applied schema change. As an optimization, YAB will not reevaluate the schema if none of the configured schema files have changed. Perhaps many months after the manual schema change was applied, a product is installed/upgraded in the environment that distributes a schema change for the same database, triggering YAB to reevaluate the schema. YAB may revert the manual change (see [Schema Changes](#) for details) and it may be difficult (in some cases) to detect that this has occurred and to understand why this has occurred.

Minimize Differences

Any change to the standard product configuration should be carefully considered. The standard product configuration is the subject of thousands of hours of testing and use and has been designed to support product upgrades without manual intervention. Overrides increase the cost of upgrades by requiring additional up front analysis and sometimes post upgrade corrections.

Segregate Differences

Having a clear separation between the standard product configuration and overrides, is valuable from the standpoint of clearly describing the environment, and in some cases, this separation can be leveraged for troubleshooting as well. The environment is organized to support this separation. The `build/config/configuration.properties` file is used to record differences from the standard product configuration and the `customizations` and `patches` directories to store customization and patch artifacts. If custom schema is required you can add the schema in a [custom database](#).

Use Comments

On occasion it may be necessary to define [configuration](#) or temporarily [skip commands](#) as part of a workaround or possibly [configuration](#) is added to reflect certain preferences. All configuration files support comments by prefixing the line with the hash (#) symbol. A simple description of the reason for the change, the user introducing the change, and the date the change was introduced, can be invaluable when evaluating upgrades and patches and understanding the background of an environment.

System Administration

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

Starting and Stopping Environments

The commands `start`, `stop`, and `status` are used to start, stop, and check the status of servers in the environment. The `restart` command is a convenience command to stop and then start the environment.

Scripts

The commands to start and stop servers typically execute a script of the same name in the scripts directory. For example, the command to start the Financials application server is `appserver-fin-start` which executes the script `scripts/appserver-fin-start`.

scripts/appserver-fin-start

```
#!/usr/bin/python
# Starts the 'fin' application server.
# Generated: 2017-03-09T10:51-0800
import os
os.environ['DLC'] = r"/qad/progress/dlc11664"
os.environ['PROMSGS'] = r"/qad/progress/dlc11664/promsgs"
os.environ['PATH'] = r"/qad/progress/dlc11664/bin:/qad/progress/java64/jdk1.7.0_45/bin:/qad/progress/dlc11664/bin:/bin:/usr/bin:/usr/local/bin:/usr/sbin"
os.environ['PROTERMCP'] = r"/qad/progress/dlc11664/protermcap"

import subprocess
import sys
params = [r"/qad/progress/dlc11664/bin/asbman", r"-i", r"as-fin", r"-start", r"-port", r"22001"]
p = subprocess.Popen(params, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
out, err = p.communicate()
print(out)
if err is not None:
    print(err)
if p.returncode != 0:
    sys.exit(p.returncode)
```

These scripts should not be directly edited. They are re-generated by executing the `script-update` (or `reconfigure` or `update`) command from the YAB console. The scripts are generated by the system so the environment can be started even if YAB has failed (and for better transparency). The YAB commands should be used in preference to direct execution of the scripts because the request will then be recorded in the YAB log file (build/logs/yab.log). Also some commands will first check the status of the managed resource before calling the script. For example not executing a script to start a server that is already running. Some servers will raise an error if they receive a request to start a server that is already running or to stop a server that is already stopped.

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	<code>dbserver.[INSTANCE].manual</code>	False
Application Server	<code>appserver.[INSTANCE].manual</code>	False
WebSpeed Server	<code>ws.[INSTANCE].manual</code>	False
Tomcat Server	<code>tomcat.[INSTANCE].manual</code>	False
Financials Daemons	<code>fin.daemons.manual</code>	False

A "manual" server can still be controlled using server specific 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
```

Controlling Financials Daemons

The Financials component is associated with background processes called *daemons*. Daemon processes are stopped after initial installation and will remain stopped until they are configured. Daemon setup is described in the *QAD System Administration User Guide*.

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.
...	

The prefix "fin.daemons" groups additional settings to fine tune how Financials daemons are managed.

```
> yab help fin.daemons
```

SETTINGS

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,ReplicationDaemon,ScanDaemon,TimeoutDaemon,CubeDaemon,ReportDaemon
fin.daemons.manual	When true the Financials daemons will not be associated with the commands:
	start stop status
commands:	But must be manually controlled and queried using the daemon-start or fin-application-start daemon-stop or fin-application-stop daemon-status See: fin.daemons.status Default: false
fin.daemons.status	When true the Financials daemons will be associated with the following command, even if 'fin.daemons.manual' is true: status See: fin.daemons.manual Default: false
fin.daemons.propath	The PROPATH to use when calling the Financials API to control daemons.
fin.daemons.startallaction	The action to submit when starting all daemons.
fin.daemons.stopallaction	The action to submit when stopping all daemons.
fin.daemons.statusallaction	The action to submit when getting the status of all daemons (currently this is not used).
fin.daemons.startaction	The action to submit when starting a specific daemon.
fin.daemons.stopaction	The action to submit when stopping a specific daemon.
fin.daemons.statusaction	The action to submit when getting the status of a specific daemon.
fin.daemons.logdir	The log directory for daemons.
fin.daemons.workdir	The work directory for daemons.

The prefix "fin.daemon" is used to configure individual daemons where the INSTANCE is the name of the daemon (e.g. eventdaemon, historydaemon,scandaemon, etc):

```
> yab help fin.daemon.INSTANCE
```

SETTINGS

fin.daemon.INSTANCE.debuglevel	The debug level for a specific daemon. The valid range is 0-31. Default: 0
fin.daemon.INSTANCE.propath	The extra daemon propath for a specific daemon.
fin.daemon.INSTANCE.extraparams	The extra daemon progress parameters for a specific daemon.

For example, to get more verbose logging from the Event Daemon define the following configuration:

```
fin.daemon.eventdaemon.debuglevel=31
```

Reconfigure and then restart the Event Daemon:

```
> daemon-configure  
> daemon-eventdaemon-stop  
> daemon-eventdaemon-start
```

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.

Schema Management

There are several commands that may be used to create and compare schema files. The [Schema Changes](#) page discusses separately the commands that are used to load schema files into a database.

Ad-hoc Commands

The following commands provide support for dumping and comparing schema files. Review the help to see how they work (yab help <command>).

Command	Description
schema-dump	Dump schema from a database.
schema-diff	Compares two schema files for differences.

Snapshot Based Commands

There are other commands that are designed to support the development and redistribution of schema (but that also may be useful in assessing the impact of installing modules into the environment). Review the help to see how they work (yab help <command>).

Command	Description
dev-schema-snapshot	Creates a "snapshot" by dumping the schema of all databases or the selected databases to a new timestamped directory.
dev-schema-diff	Compares two snapshots. The diff can be used to see pending schema changes (changes configured but not applied) as well as schema changes that are not associated with a configured schema file. Ex. Show pending schema changes. <pre>yab -configured dev-schema-snapshot yab dev-schema-snapshot yab dev-schema-list yab -snapshot1:[TIMESTAMP] -snapshot2:configured dev-schema-diff</pre> Ex. Show manually applied schema changes. <pre>yab -configured dev-schema-snapshot yab dev-schema-snapshot yab dev-schema-list yab -snapshot1:configured -snapshot2:[TIMESTAMP] dev-schema-diff</pre>
dev-schema-list	Lists the snapshots.
dev-schema-clear	Deletes all snapshots.

Snapshots will always be created after a schema change. Use the dev-schema-list command to list the snapshots in the environment.

Locating

The database-INSTANCE-schema-locate command may be used to locate the schema files that distribute schema for a table for sequence.

Ex.

```
> yab database-qaddb-schema-locate pt_mstr
Table      Active  SchemaFile
-----
pt_mstr    true    /qad/local/sandbox/cache/packages/base/2/1/0/282/qad.base/schema/progress/qaddb
/qaddb.df
pt_mstr    false   /qad/local/sandbox/cache/packages/mfg/2017/0/17/536/schema/qaddb/mfgempty.df
```

Data Management

Environment Backup

The `environment-offline-backup` and the `environment-online-backup` commands are used to perform an environment-wide backup of all system components. The offline command stops the environment before taking the backup whereas the online command does not, though it may temporarily take components offline that do not support an online backup. The offline backup is the recommended approach to ensure data consistency across components.

```
> yab environment-offline-backup
```

Backups are located in subdirectories of the directory configured by the `dbbackup.dir` setting. When `dbbackup.timestamp` is true, the subdirectory names encode the time of the backup using the format `([YEAR][MONTH][DAY][HOUR][MINUTE][SECOND])`. The `dbbackup.timestampmax` setting can be configured to specify a maximum number of backups to retain, where the oldest backup will be discarded to make way for a new backup. This ensures backups do not accumulate. When `dbbackup.timestamp` is false, the name of the directory is controlled using the `(-tag:NAME)` option, where `NAME` is the name of the subdirectory where the backups should be written. If the `(-tag)` option is not defined, backups are written to a subdirectory named "default." If the subdirectory contains a backup for the component, it will be overwritten.

Progress Databases

The Progress databases are included in the environment backup. Additional "dbbackup" settings can be used to fine tune the configuration of the Progress database backups, as well as process-level options for enabling after imaging and replication:

```
> yab help dbbackup
> yab help database-backup
```

Compression

Database compression is enabled by default:

```
dbbackup.compress=true
```

On non-Windows platforms, the backup is compressed as it is written. This compression conserves disk space but increases the time it takes to back up the database, including the time required to back up the before image and the time that database operations are suspended when the system is online. The compression can be delayed until after the backup has completed to reduce this delay but requires additional disk space usage.

```
dbbackup.delaycompression=true
```

Compression can be completely disabled:

```
dbbackup.compress=false
```

Environment Restore

The `environment-restore` command is used to restore all of the components in a backup. The command stops the environment before restoring the backup.

Backups are located in subdirectories of the directory configured by the `dbbackup.dir` setting. To restore a specific backup, choose the appropriate subdirectory name using the `(-tag:NAME)` option, where `NAME` is the name of the backup subdirectory to restore. If the `(-tag)` option is not defined, the backup in a subdirectory named "default" will be restored if it exists, otherwise an error will be raised. A backup may include data for multiple components and the `environment-restore` will restore all of the components. The `environment-restore` process may also include additional steps to ensure the restored components are synchronized to a consistent state. Use component-specific restore commands to restore individual components (e.g. `database-qadadm-restore`).

The `database-all-backup-list` command shows the components included in each backup.



The following one-liner prints the backup subdirectory names:

```
yab -q config dbbackup.dir | cut -d'=' -f2 | xargs ls -l
```

Progress Databases

When [After Imaging](#) is enabled on a database and the Progress AI archiver is used to archive AI extents, archived AI files will be moved from the directory configured by `dbbackup.INSTANCE.archivaldir` to the backup to which they apply. This can be disabled with the following setting:

```
dbbackup.movearchivedaifiles=false
```

YAB does not provide any commands for after imaging "roll-forward" recovery. Consult the Progress documentation for instructions on restoring AI files.



If database security is enabled (see `'db.INSTANCE.security.authentication'`), use `database-security-reapply` to reapply the security configuration in the restored databases. When "roll forward" recovery is used, roll forward the [After Imaging](#) files before reapplying the security configuration to avoid invalidating the after-imaging files.

```
yab -tag:NAME database-security-reapply
```

Configuration Backups

The system automatically creates backups of the configuration when it changes and passes validation. The backups are stored in the work directory:

```
> ls -lt build/work/system/config/
total 80
drwxr-xr-x. 3 wtb devel 4096 Mar 26 11:43 current/
drwxr-xr-x. 3 wtb devel 4096 Mar 26 11:43 2018-03-26T11-43-36/
drwxr-xr-x. 3 wtb devel 4096 Mar 26 11:39 2018-03-26T11-39-20/
drwxr-xr-x. 3 wtb devel 4096 Mar 22 12:36 2018-03-22T12-36-39/
drwxr-xr-x. 3 wtb devel 4096 Mar 22 11:44 2018-03-22T11-44-46/
drwxr-xr-x. 3 wtb devel 4096 Mar 22 08:21 2018-03-22T08-21-03/
drwxr-xr-x. 3 wtb devel 4096 Mar 13 11:25 2018-03-13T11-25-42/
drwxr-xr-x. 3 wtb devel 4096 Mar 12 15:36 applied/
drwxr-xr-x. 3 wtb devel 4096 Mar 12 15:36 2018-03-12T15-36-31/
```

The "current" backup represents the last known good configuration. When a new backup is created, the files in the "current" backup are moved into a timestamp directory to make way for the new backup. The "applied" backup stores the configuration of the system during the last successful [update](#). The `system-config-list` command provides a convenient way to list all of the backups (excluding "current" and "applied").

```
> yab system-config-backup-list
2018-03-26T11-43-36
2018-03-26T11-39-20
2018-03-22T12-36-39
2018-03-22T11-44-46
2018-03-22T08-21-03
2018-03-13T11-25-42
2018-03-12T15-36-31
```

The `system-config-restore` command can be used to restore a configuration backup, but it is not useful if the configuration is so broken that a YAB session cannot be started. For this reason, it is recommended that backups should be restored using the new YAB client startup option (`-config-restore`) which can repair the configuration before starting a YAB session.

When used without a value, it will restore the last known good configuration backup (current).

```
> yab -config-restore
```

Alternatively a specific backup to restore can be specified by name:

```
> yab -config-restore:2018-03-26T11-43-36
```

If any configuration files are updated or removed by restoring the backup, copies of the files are stored in an archive in the current working directory.

```
> yab -v -config-restore:2018-03-26T11-43-36
...
2018-03-26 12:24:21,435 DEBUG [main] Main - Restored [/qad/sbox/004/user/wtb/01/sm1/build/work
/system/config/2018-03-26T11-43-36/system/loader.properties] to [/qad/sbox/004/user/wtb/01/sm1
/build/config/system/loader.properties].
2018-03-26 12:24:21,454 DEBUG [main] Main - Restored the configuration from [/qad/sbox/004/user
/wtb/01/sm1/build/work/system/config/2018-03-26T11-43-36] and archived modified and removed
configuration files to [config-20180326122421.zip].
```

Data (Native Format)

Overview

Data that is dumped using the Progress Data Dictionary in text (*.d) or binary (*.bd) format can be loaded into the environment using the data configuration type.

Loading Data

Loading a set of .d format data files is as easy as selecting the data files to load and the destination database.

Ex. Configure the environment to load the .d data files in the test directory.

build/config/configuration.properties
<pre>data.test.dir=/qad/sandbox/user/wtb/02/sc2-whole/test data.test.includes=*.d data.test.database=db.qadadm data.test.format=dotd</pre>

With the preceding configuration, the .d data files would be processed during the database-qadadm-data-dotd-update step in an update.

To load binary format data set the binary setting on the data instance to true.

build/config/configuration.properties
<pre>data.test.binary=true</pre>

Data (XML Format)

Overview

Traditionally Progress data is dumped to the file system using the Progress Data Dictionary in text (*.d) or binary (*.bd) format. These "DOTD" files can be loaded back into the same environment or into another environment using the Progress Data Dictionary again to do the work. Progress also supports dumping and loading data in XML format by leveraging ProDataSets. Whereas the DOTD data format is still the format of choice for large data sets, redistributing data in XML format has certain advantages for small and medium size data sets.

Advantages of XML Data

Update/Delete Support

A DOTD file can only be used to add records to the database. If a record exists, the Progress Data Dictionary will raise an error and subsequent records in the file will not be processed (unless the error threshold is configured to ignore these errors). Likewise there is no way to remove a set of records using the Progress Data Dictionary. In YAB an XML data file can be processed in a mode where it will add records that do not exist and update records that do exist, a mode where they will be added but not updated, and a mode where the corresponding records will be removed from the database if they exist.

Portability

XML data files are always dumped/loaded using a standard encoding (UTF-8). Date values and decimal values are represented using standard XML Schema Definition types, so there is no concern over differences in the Progress startup parameters (mdy, yy, numsep, numdec) used to dump and to load the data.

Flexibility

The Progress Data Dictionary dumps all records and all fields from a single table into a DOTD file. In YAB it is possible to join tables (e.g Order to Order Lines), filter records, and select fields to dump into an XML data file.

Human Readable

XML data files are easier to read and edit because the document format includes schema information whereas the dotd format is positional.

Diff/Patch Support

YAB provides support for comparing XML data files to show differences in terms of record adds and deletes and field updates (diff). These differences can be merged into another XML data file (patch).

Loading Data

The data configuration type is used to configure data to be loaded into an environment. This type can be used to load XML as well as binary and text dotd files. Loading a set of XML data files produced in a YAB environment is as easy as selecting the data files to load and the destination database.

Ex. Configure the environment to load the XML data files in the test directory.

build/config/configuration.properties

```
data.test.dir=/qad/sandbox/user/wtb/02/sc2-whole/test
data.test.includes=*.xml
data.test.database=db.qadadm
data.test.format=xml
```

With the preceding configuration, the XML data files would be processed during the database-qadadm-data-xml-update step in an update.

Managing Data

YAB provides the following commands for working directly with XML data. Review the help to see how they work (yab help <command>).

Command	Description
data-xml-dump	Dumps data from a database into XML data files.

data-xml-load	Loads XML data files into a database.
data-xml-diff	Compares two data files for differences.
data-xml-patch	Applies a diff to a data file.

Managing System Data

YAB also provides a set of commands that are designed to support the development and redistribution of system data. The definition of system data entities is configured by the `dev-data` configuration type and comes pre-configured with support for the standard system data entities.

Command	Description
dev-data-snapshot	Creates a "snapshot" by dumping all system entities to a new timestamped directory.
dev-data-diff	Compares the most recent snapshot to the database or any two snapshots.
dev-data-patch	Compares the most recent snapshot to the database or any two snapshots and applies the differences to a directory of data files.
dev-data-list	Lists the snapshots.
dev-data-clear	Deletes all snapshots.
dev-data-entity-list	Lists the configured data entities.

To add support for an additional table/entity create a new dump request (see the help for `data-xml-dump`) and then append the dump request to the configured system data entities.

Ex. Adding support to dump records from the table `chui_det`.

dump-chui_det.xml

```
<Entity>
  <Dataset>
    <DatasetName>chui_det</DatasetName>
    <Table>
      <TableName>chui_det</TableName>
    </Table>
  </Dataset>
</Entity>
```

build/config/configuration.properties

```
# @append
dev-data.databases.qadadm=/qad/sandbox/user/wtb/ee/dump-chui_det.xml
```

Compiling Source Code

Progress source code is compiled using *code* commands where the INSTANCE identifies a specific compile configuration:

code-INSTANCE-update



To list the compile instances:

```
yab -instances config code
```

Ex. Compile standard operational source code:

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

Ex. Compile customized operational source code:

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

The compile is an *incremental compile* which means files will only be recompiled if they have changed, or a resource on which they depend, has changed. When a file is recompiled a message is written to the log explaining why the file was recompiled.



The incremental compile uses Progress compile XREF files to understand program dependencies, however Progress does not support generating XREF files for encrypted programs. The default configuration will recompile a file if it is encrypted or if it depends on a file that is encrypted. This strict approach to correctness ensures that all source code changes are processed. (Consider an example where the unencrypted program A includes the encrypted file B. We know if A or B has changed, but we do not know that B depends on the changed file C, because we cannot get dependency information for B.)

See [Compile for Developers](#) for techniques to streamline compiles in development environments.

You can force a full recompile using the (-force) option:

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

The (-nocompile) option can be used to see what programs would be compiled without actually compiling the programs:

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

Configuring Compiles

Progress source code compilation is configured through instances of the *code* configuration type:

`code`. INSTANCE

Ex. Configuration of the standard operational compile:

```
> yab config "code.mfg.*"
code.mfg.databases=db.qadcpl,db.qadadm,db.qadhl,db.qadrcode
code.mfg.defaultincludes=**/*.p,**/*.w,**/*.t,**/*.cls
code.mfg.dir=/qad/sbox/001/user/wtb/rfrd-4117/dist/mfg
code.mfg.failonerror=true
code.mfg.languages=us
code.mfg.params=-T /tmp -d mdy -yy 1950 -s 32768 -mmax 8192 -inp 32000 -rereadnolock -c 30 -D
1000 -Bt 350 -nb 200 -h 25 -tok 4096 -tmpbsize 8 -TB 31 -TM 32 -cpstream utf-8 -cpinternal utf-8 -
cprcodeout utf-8 -cpcoll ICU-UCA -cpcase basic
code.mfg.propath=/qad/sbox/001/user/wtb/rfrd-4117/patches/fin/proxypatch,/qad/sbox/001/user/wtb
/rfrd-4117/patches/mfg/default/src,/qad/sbox/001/user/wtb/rfrd-4117/build/work/archiving/mfg
/progress/xrc,/qad/local/sandbox/cache/packages/mfg-ee2016-patch/2016/0/16/440/src,/qad/local
/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src,/qad/local/sandbox/cache/packages/mfg-ee2016
/2016/0/16/209/src/validation,/qad/local/sandbox/cache/packages/base-api/1/1/0/79/src,/qad/local
/sandbox/cache/packages/mfgcoreplus-api/2/17/0/15/src,/qad/local/sandbox/cache/packages/qracore-
api/2/17/0/32/src,/qad/local/sandbox/cache/packages/requisition-api/1/1/0/91/src,/qad/local
/sandbox/cache/packages/fin-src-proxy/2016/0/80/18
code.mfg.removeorphanedrcode=true
code.mfg.sources.archiving.dir=/qad/sbox/001/user/wtb/rfrd-4117/build/work/archiving/mfg/progress
/xrc
code.mfg.sources.main.dir=/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src
code.mfg.sources.main.excludes=**/gpsc02.p,**/urltempl.p,**/utdznc.p,**/wbqu01.p,**/wbqu02.p,**
/wbqu03.p,**/utsequp1.p,**/lvcntora.p,**/gpbrncha.p
code.mfg.sources.main.extra.excludes=encoding.dat,locale.dat
code.mfg.sources.main.extra.includes=*.dat,*.ini,version.mfg,**/*.i
code.mfg.sources.mfg-patch.dir=/qad/local/sandbox/cache/packages/mfg-ee2016-patch/2016/0/16/440
/src
code.mfg.sources.mfg-patch.excludes=**/gpsc02.p,**/urltempl.p,**/utdznc.p,**/wbqu01.p,**/wbqu02.
p,**/wbqu03.p,**/utsequp1.p,**/lvcntora.p,**/gpbrncha.p
code.mfg.sources.patches-default.dir=/qad/sbox/001/user/wtb/rfrd-4117/patches/mfg/default/src
code.mfg.threads=1
```

To print the documentation for all *code* settings:

```
> yab help "code."
```

Compile for Developers

The default compile configuration chooses strict correctness over efficiency. In development environments, the default configuration can slow down iterative development as developers wait for compiles. This penalty is much greater when compiling source code that is encrypted or partially encrypted. In development environments, the balance between correctness and efficiency can be adjusted to boost developer productivity, while also providing the ability to more completely validate a change at the end of a development cycle before promoting the change to non-development environments.

Enabling Developer Optimizations

The dev setting on the code instance is used to enable developer optimizations.

Ex. Configure the compile of customized operational source code for development.

```
code.mfg-customizations.dev=true
```

The dev setting is a "convenience" setting that implies the following settings (see yab help "code." for details on each of these settings) :

```
forceencrypted=false
scan=false
removeorphanedrcode=false
skipbeforeimagertruncation=true
nometa=true
```

Developers on multi-core servers might also experiment with the threads setting to see if it improves performance.



The dev option disables the incremental compile (nometa=true). The compile will recompile ALL source code unless the scope of the compile is defined (see below).

Controlling Compile Scope

The developer has two options to control the scope of a compile.

Single Source Compile

A compile is associated with one or more "sources" (see yab -instances config code.INSTANCE.sources). Commands are provided to compile each source individually.

Ex. List the compile commands associated with the customized operational source code compile:

```
> tree -d -L 2 customizations/mfg/
customizations/mfg/
|-- another |
|-- data |
    |-- src
    |-- default
|-- data
    |-- src

> yab -all help code-mfg-customizations

PROCESSES

    code-mfg-customizations-customizations-default-update    Recompiles 'mfg-customizations:
customizations-default' source code.
    code-mfg-customizations-locate                            Locates files associated with a
compile.
    code-mfg-customizations-rebuild                            Removes and recompiles all 'mfg-
customizations' code.
    code-mfg-customizations-remove                            Removes all 'mfg-customizations'
compiled code.
    code-mfg-customizations-update                             Recompiles all 'mfg-
customizations' source code.
```

Following the previous example, if the "default" source and the "another" source do not contain overlapping files then the following commands can be used to limit the scope of the compile to either source:

Source	Command
default	code-mfg-customizations-customizations-default-update
another	code-mfg-customizations-customizations-another-update



These compile commands can be quite long. You can define an "alias" for any command to shorten the name.

build/config/configuration.properties

```
process.alias1.id=cpld
process.alias1.depends=code-mfg-customizations-customizations-default-update
process.alias2.id=cpla
process.alias2.depends=code-mfg-customizations-customizations-another-update
```

```
> yab cpld
```

```
                cpld (1 task)
```

```
-----
1/1 code-mfg-customizations-customizations-default-update  OK (0.406 s)
-----
```

```
BUILD SUCCESSFUL (9.373 s)
```



The Progress compile must be invoked using PROPATH relative paths to locate the files to compile (compiling with absolute paths produces different results). A single source compile may reference a file outside of the source if another source has a different version of the file and is higher on the compile PROPATH.

Compile List

A compile list may be provided to define the scope of the compile using the `devlist` setting.

The compile list may be configured:

```
code.mfg-customizations.devlist=${customizations.mfg.dir}/dev-compile.lst
```

Or defined on a per request basis:

```
yab -code.mfg-customizations.devlist:dev-compile.lst code-mfg-customizations-update
```

The compile list should enumerate the source code files to compile (one per line) using relative paths resolved against the compile PROPATH.



When the `devlist` setting is defined, the `compilelist` and `sources` settings associated with the compile are ignored.

Validating Changes

Changes that are validated with developer optimizations enabled should be re-verified with developer optimizations disabled and without any limitations on the scope of the compile. This validation ensures that the state of the system can be reliably reproduced in non-development environments and is not dependent on partial or incomplete compilation of the source code. If the `devlist` setting was configured it should be disabled for this verification.

Ex. Compile the customized operational source code with development optimizations disabled.

```
> yab -code.mfg-customizations.dev:false code-mfg-customizations-update
```

Locating Compile Resources

Every compile has a locate command to locate compile resources. The command lists resources alphabetically ordered by their relative path from the compile propath and secondarily in the order in which they are located on the compile propath. Files with the same relative path (overrides) are listed together, and in the case of source code files, the first file listed is the file that will be compiled.

Ex. Locate all standard/patched operational source code files.

```
> yab code-mfg-locate
File
Modified          Hash                               Enc
-----
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/applhelp.
p                               2016-02-26 02:01 AM
ed641e4af8dc9beadb95ee3653a15e No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/data/BufferFieldMap.
cls                               2016-02-26 02:01 AM d8b01e1b613831dcba3e7a1023f97 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/data/BufferMap.
cls                               2016-02-26 02:01 AM 7db1375cb2cab19ca5e370ed3b32f77 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/data/CamelCaseMap.
cls                               2016-02-26 02:01 AM 3022bd84e174b96c195c22ad3e13db7 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/data/DatasetFieldMap.
cls                               2016-02-26 02:01 AM dd355c363bbe811e57e71a05b44268e No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/data/IBufferFieldMap.
cls                               2016-02-26 02:01 AM 55e41a7edd8d4ddf2dcb244e82ae1f8 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/data/IBufferMap.
cls                               2016-02-26 02:01 AM 3d70a402d24d218b650db70b92da848 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/data/IDatasetFieldMap.
cls                               2016-02-26 02:01 AM 58189519c8294e7d5122a45f9e57633a No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/data/PassThruFieldMap.
cls                               2016-02-26 02:01 AM c230af26d9fab58d98a0f529e4cb542 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/erp/Logger.
cls                               2016-02-26 02:01 AM
687f329856a9a11e3257cb83a05a70e3 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/com/qad/erp/api/APIDispatcher.
p                               2016-02-26 02:01 AM 4c92a1f73780cf8dbb1243a18b22f31 No
...
```

Ex. List all standard/patched versions of the operational file 'us/lv/lvgenpl.p'.

```
> yab code-mfg-locate us/lv/lvgenpl.p
File
Modified          Hash                               Enc
-----
/qad/local/sandbox/cache/packages/mfg-ee2016-patch/2016/0/16/390/src/us/lv/lvgenpl.p 2016-08-15
03:00 PM 70dcebd43f54a6a87d1c309ce34b5c No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/us/lv/lvgenpl.p 2016-02-26
02:01 AM efedbbd6be2539ee85de2317f77d4543 No
```

Ex. Compare files where there is a mixture of encrypted and unencrypted versions.

```
> yab -xcode code-mfg-customizations-locate us/gp/gpumxr.p
File
Modified          Hash (X)                               Enc
-----
/qad/sandbox/user/wtb/02/sc2/customizations/mfg/default/src/us/gp/gpumxr.p 2016-09-07 09:16
AM f550e03f398f91865eeac72432cf4c Yes
/qad/local/sandbox/cache/packages/qgrabridge-erp/1/2/0/60010/src/us/gp/gpumxr.p 2016-08-18 05:01
AM f550e03f398f91865eeac72432cf4c No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/us/gp/gpumxr.p 2016-02-26 02:01
AM 9e5aa3b1570e18a9e29d63fb23c8259 No
```

Ex. List all overridden standard/patched operational source code files.

```
> yab -overrides code-mfg-locate
File
Modified          Hash (X)                      Enc
-----
-----
/qad/local/sandbox/cache/packages/qrabridge-erp/1/2/0/60010/src/mfaspl.p      2016-08-18
05:01 AM 1ebae0fda23174c011f158b8b3f55ab No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/mfaspl.p      2016-02-26
02:01 AM 1890d46f68d68fc11d81754dd7461e4e No
/qad/local/sandbox/cache/packages/qrabridge-erp/1/2/0/60010/src/mfinitpl.p    2016-08-18
05:01 AM ff7090368537cec049e61b111fd77 No
/qad/local/sandbox/cache/packages/mfg-ee2016-patch/2016/0/16/390/src/mfinitpl.p 2016-08-15
03:00 PM c343d68da2b5e82a43892add57614f4 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/mfinitpl.p    2016-02-26
02:01 AM eb1277faca8f9e7c2b0a6c51abf1e5b No
/qad/local/sandbox/cache/packages/qrabridge-erp/1/2/0/60010/src/mflg01.p      2016-08-18
05:01 AM fb76dfb98491c9d9e776855b7c2b No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/mflg01.p      2016-02-26
02:01 AM a32453afead3b5f47918648844825484 No
/qad/local/sandbox/cache/packages/qrabridge-erp/1/2/0/60010/src/mfssmsg.p      2016-08-18
05:01 AM 5425640156b5b6860bb575a9a24686d No
/qad/local/sandbox/cache/packages/mfg-ee2016-patch/2016/0/16/390/src/mfssmsg.p 2016-08-15
03:00 PM d03ebf11851b7f2cfdc4b8691dbaf619 No
/qad/local/sandbox/cache/packages/mfg-ee2016/2016/0/16/209/src/mfssmsg.p      2016-02-26
02:01 AM 57b3b2a79c1a51465460d55ef839de No
...
```

Ex. List overridden standard/patched operational source code files, where the overriding file (the file that has won) is not distributed in a package.

```
> yab code-mfg-locate -overrides -skip-packages
File
Modified          Hash (X)                      Enc
-----
-----
/dr01/qadapps/sm2/patches/mfg/default/src/us/wo/wotrantt.i
i
ffe51d3e7d8297237588704eeddc6ab2 No 2019-01-14 11:53 AM
/dr01/qadapps/sm2/build/catalog/packages/pushproduction/1/2/0/189/qad.pushproduction/code/progress
/mfg/us/wo/wotrantt.i 2018-09-25 03:09 PM 8a214e78837878e76e395ec6eb2dd No
/dr01/qadapps/sm2/build/catalog/packages/mfg/2018/0/18/711/src/us/wo/wotrantt.i
i
2015-01-16 12:57 PM 82af8ce259d671fd9e17cdf2eebb50
No
```

Ex. Record a specific override as resolved.

```
> yab -resolve code-mfg-customizations-locate us/gp/gpumxr.p
```

Ex. Record all overrides as resolved.

```
> yab -resolve code-mfg-customizations-locate
```

The command help has further information.

```
> yab help code-mfg-locate
```

Generating XREF output

Progress XREF files in XML format are produced when a program is compiled to support the incremental compiler. Normally the XREF files are deleted after they are processed to conserve on disk space usage. The `keepxref` setting instructs the compiler to not delete the XREF files.

Ex. Retain XREF files generated during the operational compile.

```
build/config/configuration.properties
```

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

Ex. Recompile all operational code to force the generation of XREF files for all programs.

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

The XREF files are generated into the directory where the rcode is written.

```
[RCODE DIRECTORY]/.meta/xref
```

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

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



XREF output will only be generated for programs that are not encrypted.

Generating DEBUG-LIST output

You can configure DEBUG-LIST output to be generated when a specific set of programs are compiled or when all programs are compiled.

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

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

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

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

Ex. Configure DEBUG-LIST output for all operational programs.

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

Ex. Compile the operational code

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

The DEBUG-LIST files will be generated into the directory where the rcode is written.

[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 programs that are not encrypted. All selected programs will be recompiled (even if they have not changed.)

Compile Databases

The *db.qadcpl* and *db.qadrcode* databases have a special role in some compiles.

Compile Database (db.qadcpl)

The *db.qadcpl* database has the same schema as the *db.qaddb* database, but without any tables or sequences belonging to the QAD Reference Architecture (QRA). Compiles connect the *db.qadcpl* database instead of the *db.qaddb* database to ensure that programs use QRA APIs to access QRA data, and not direct table access. Additionally, the compile database avoids (ambiguous reference) compile errors that may occur when compiling legacy programs with unqualified references to tables, indexes, and fields.



The system will automatically mirror schema configured for the *db.qaddb* database to the *db.qadcpl* database. To prevent this mirroring set "schema.INSTANCE.qra=true".

RCode Database (db.qadrcode)

The r-code database is part of a solution to create a single set of r-code to support all languages, instead of separate r-code for each language.

Auditing

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

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

QAD Security and Controls Section	Notes
Enabling Auditing for the Database	The databases are all pre-configured with database areas 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 <i>QAD Security and Controls</i> documentation, but the location of the audit policy files has changed. See Importing Policy Files for 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	See Disable Auditing to disable auditing on databases.
Loading Electronic Signature Initial Data	The data files to load with program E-Sig Initial Data Load (36.12.14.13) are available in the <i>config/electronic-signature</i> directory.

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
```

By default non-primary indexes on auditing tables will be deactivated on source databases and activated on archive databases (databases whose configuration contains "qadarc" as in "db.qadarc"), to improve performance and limit the growth of storage areas used by audit data, while supporting reporting on the audit data in the archive database. This default can be explicitly overridden with the option (-deactivateidx or -deactivateidx:false). The system will use storage areas named AUDIT_DATA and AUDIT_IDX to store the auditing data and indexes when auditing is enabled. When a database is created by YAB these storage area names are automatically added to the structure of the database. Alternative storage areas may be used by configuring the areas for each database (and ensuring the database has the storage areas defined using Progress *prostrct add*).

```
db.qaddb.audit.area=AUDIT_DATA  
db.qaddb.audit.indexarea=AUDIT_IDX
```



The database must be offline to enable auditing.

Disable Auditing

To disable auditing on all databases, execute the command:

```
> yab database-auditing-disable
```

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

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



The database must be offline to disable auditing.

Importing Policy Files

The *Importing Audit Policy* section in the *QAD Security* 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](#). The archive will export all audit data in the source database, load the exported data into the archive database, and then delete the exported data from the source database.

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
```

Configuration settings that begin with "audit" are used to configure how audit records are handled.

```
> yab help audit

SETTINGS

    audit.archive.database    The database where archived audit records will be stored.

    audit.archive.dir        The directory where exported audit records will be stored.

    audit.checkseal          True to verify that the seal of each audit data record matches
the database MAC key prior to writing it to the audit archive file.
    This option requires that the audit policies in effect when
writing the audit data records have specified a Data Security Level of DB Passkey.
    For more information on creating audit policies and data
security, see OpenEdge Getting Started: Core Business Services - Security and Auditing.

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

    audit.user               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.validate.indexes   When true checks the auditing indexes in the source and archive
database for corruption and attempts to fix the corruption if any is detected.
```

The command help includes these options as well as a few additional options which typically would be defined as part of a specific request (e.g. -audit.daterange:"09-07-2005 17:00:00.000-04:00").

```
> yab help database-qaddb-auditing-archive
```

PROCESS

database-qaddb-auditing-archive - Moves audit records in the 'qaddb' database to the archive database.

OPTIONS

Name	Description
-----	-----
audit.daterange	Used to limit the records that will be archived.
	If one date is specified, all audit records in the database with a date equal to or earlier than the specified date will be archived.
	If two dates are specified (separated by a space), all audit records with a date between the two dates specified, inclusively, will be archived. If a date range is not specified, the system will archive all audit records equal to or earlier than the current date and time.
	Dates must be specified using the format "mm-dd-yyyy hh:mm:ss.sss+hh:mm". For example, to specify 5:00 P.M. on September 7, 2005 EDT, enter "09-07-2005 17:00:00.000-04:00".
audit.recs	Specifies the number of records to delete in a single transaction. The archive executes a transaction for every num-recs records it deletes. The default for num-recs is 100,
	and the maximum is your current Lock Table Entries (-L) setting. The multi-user default for -L is 8192.
...	



If the `audit.user` was not defined when the environment was created, the operating system account that was used to install the environment, would need to be setup as an application 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](#).

To manually start the archive database server:

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

To manually stop the archive database server:

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

This default can be changed with the following configuration setting:

build/config/configuration.properties
dbserver.qadarc.manual=false

QXtend

Reapply QXtend Defaults

An environment is created with the QXtend default configuration. Afterwards specific settings like application server ports may be adjusted during an [update](#) but the default configuration is not reapplied because reapplying the default configuration would overwrite any subsequent configuration.

To force the re-application of the default configuration:

```
> yab -qxtend.reconfigure start qxo-services-stop qxtend-default-configuration
```

Enterprise Edition Best Practices

The following best practices promote the health of an Enterprise Edition installation.

- [Trim Application Servers](#)

Trim Application Servers

Over time, application server agents may become unavailable. When this occurs, the performance of the system is degraded as fewer agents are available to service the workload. A good practice is to schedule a task to periodically trim and replace application server agents.

Ex. Trim and replace 25% of the configured agents.

```
> yab -trim-percent:25 -trim-replace appserver-trim
```

For help on trim options:

```
> yab help appserver-trim
```

Database Security Management

- [Enabling Database Authentication](#)
- [Enabling Database Table Level Permission](#)

Enabling Database Authentication

This section describes how to enable Progress database authentication.

Configuration

By default, database authentication is not enabled in YAB. To enable database authentication, configure the db.INSTANCE.security.authentication setting and corresponding connection user settings. The following configuration will define connection users for all of the databases and also enable database authentication.

```
db._base.security.authentication=true
db._base.connection.user=qadadmin
db._base.connection.password=qad
db._base.connection.dba=true
db._base.connection.managed=true
```

To configure other database users, please refer to the following configuration:

```
db._base.users.type1.user=test1
db._base.users.type1.password=qad
db._base.users.type1.dba=false
db._base.users.type1.managed=true

db._base.users.type2.user=test2
db._base.users.type2.password=qad
db._base.users.type2.dba=false
db._base.users.type2.managed=true
```

To apply these settings to the current environment, execute the command:

```
> yab reconfigure
```

Help

For detailed usage of database authentication settings, refer to the YAB help:

```
> yab help db.*.security
> yab help db.*.connection
```

And for other related database security processes, execute the command:

```
> yab help database-security
```

Enabling Database Table Level Permission

This section describes how to enable Progress database table-level permissions.

Configuration

To enable Progress database table-level permissions, YAB provides the following properties to set up a series of rules to control which users/tables/permissions can be accessed or denied.

```
db.INSTANCE.table-auth.NAME.includes=  
db.INSTANCE.table-auth.NAME.excludes=  
db.INSTANCE.table-auth.NAME.users=  
db.INSTANCE.table-auth.NAME.permissions=
```

The following example defines a rule that gives users 'mfg' and 'test*' access to tables with the suffixes '_mstr' and '_det' but not to tables with the prefixes 'fhd_' and 'lbs_' and provides read and create permission for the selected tables.

```
db._base.table-auth.type1.includes=*_mstr,*_det  
db._base.table-auth.type1.excludes=fhd_*,lbs_*  
db._base.table-auth.type1.users=mfg,test*  
db._base.table-auth.type1.permissions=can-read,can-create
```

To apply the settings to the current YAB environment, execute the command:

```
> yab database-security-table-permission-update
```

Help

For detailed information on table permission settings, refer to the YAB help:

```
> yab help db.*.table-auth
```

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-customizations-update

[Compiles](#) the operational source code customizations.

Displays the configuration of the operational customizations compile:

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

code-mfg-update

[Compiles](#) the standard operational source code including patches.

Displays the configuration of the compile:

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

config

The *config* command is used to query configuration settings.

```
> yab config adminserver.adminport
adminserver.adminport=22092

> yab config | more
adminserver.adminport=22092
adminserver.conmgr=/dr01/qadapps/qa/build/work/generated/conmgr.properties
adminserver.plugins=/dr01/qadapps/qa/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/qa/build/work/generated/ubroker.properties
aia._base.controllingnameserver=ns-default
aia._base.host=vmdvr02
aia._base.logfile=/dr01/qadapps/qa/build/logs/.log
aia._base.logginglevel=3
aia.default.allowaiacmds=1
aia.default.context=aia
...

> yab config | grep ssh
netui.connmgr.protocol=ssh
netui.protocol=ssh
netui.telnet.protocol=ssh
qxtend.connmgr.protocol=ssh
```

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

```
> yab -trace config languages
languages

[1: build/config/configuration.properties]
languages=us,ge

[2: build/config/packages/yab-ee-foundation/1/5/0/16/important/yab-ee-foundation.properties]
languages=us
```

The (-listorder) option is useful to order the trace output of settings where the value is a list of items (i.e. PROPATH).

The (-instances) option is used to print the instances of a configuration type.

```
> yab -instances config db
db.qadadm
db.qadarc
db.qadcpl
db.qaddb
db.qadhlp
db.qadrcode
db.qxevents
db.qxodb

> yab -instances config appserver
appserver.fin
appserver.mfg
appserver.qra
appserver.qxosi
appserver.qxoui
appserver.qxtnative
```

The search phrase may contain a "*" wildcard to match zero or more characters (and it is generally a good idea to put the phrase in quotes to prevent the shell from interpreting the wildcard character). Here are some additional commands for frequently queried resources:

Command	Description
---------	-------------

Proprietary of QAD, Inc.

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 configured.
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.

```
> yab -instances config dbdb.qadadmdb.qadarcdb.qadcpldb.qadddb.qadhlpdb.qadrcodedb.qxeventsdb.qxodb
```

The (-show-validation) option is used to list the validation rule that applies to the configuration.

```
> yab -show-validation config code.mfg.dir
code.mfg.dir=/qad/sbox/004/user/sji/01/whole2017/dist/mfg
(code.mfg.dir: req)

> yab -show-validation config aia.*.logfile
aia._base.logfile=/qad/sbox/004/user/sji/01/whole2017/build/logs/${name}.log

aia.default.logfile=/qad/sbox/004/user/sji/01/whole2017/build/logs/aia-default.log
(aia.default.logfile: req)    (?aia.default.logfile: parentdir)
```

database-storage-area-resolve

Introduction

The command `database-<instance>-storage-area-resolve` can provide user an approach to check whether the storage areas assigned to progress database objects matches the configuration and optionally resolve the inconsistencies.

This command can resolve table, index and CLOB/BLOB field storage area inconsistencies between configured schema definition and actual installed database. It will apply the specified configured storage area definition to the installed database after executing with `-resolve` option.

The table and index storage area are defined in schema definition file(.df) as "AREA <STORAGE_AREA>" under the table or index definition.

The CLOB/BLOB fields storage area are defined in schema definition file(.df) as "LOB-AREA <STORAGE_AREA>" under the field definition.

Furthermore for the storage area, you can find the definitions in the structure file(.st).

Storage Object Type	Need Dump /Load	Comment
table	no	The table storage area inconsistency can be resolved both online and offline.
index	no	The index storage area inconsistency can be resolved both online and offline
CLOB/BLOB field	yes	The CLOB/BLOB field storage area inconsistency must be resolved with offline

For the processing flow, the process will use the existing schema what is in use to create a temporary database(just structure without data loading). Then to use the progress query to compare temporary database and the installed database to list all the inconsistencies as a result.

If the `-resolve` option is explicitly defined in the command, the process will continue to resolve the inconsistencies as the following processing steps.

For the case resolving CLOB/BLOB fields storage area, it needs dump the current data in the database first. Then drop the table which contains the CLOB/BLOB fields. The schema update process will recreate all the table includes storage area changes after dropping the target table. After the recreation, load back the backup data. Since this scenario needs to dump/drop/load, offline is pre-requisite.

For the case resolving only table storage area, the process will use progress command ***tablemove*** to applied the storage area changes.

For the case resolving only index storage area, the process will use progress command ***idxmove*** to applied the storage area changes.

For the hybrid case(contains both table and index area change), it depends on the target index area is single or multiple. If the target index area is single, we can apply both table and index storage area changes in single one ***tablemove*** command instead of processing them separately. If the target index area is multiple, we need to apply all the indexes changes one by one first, then apply table storage area changes after the index area changes applied.

For the detail logic, please see the flowchart in the [appendix](#) as below.

Options

Name	Description
-----	-----
resolve	Correct the inconsistencies listed in report. Default: false
tables	A comma-delimited list of tables to evaluate and optionally correct. Default: all tables Ex. table1,table2

Usage & Examples

Default

Option	Default value	Description
resolve	false	Only report the inconsistencies. For resolving the inconsistencies, explicitly specified is necessary.
tables	all tables	-

Examples

The following examples are using qadhelp as a sample for instance.



In the output report, "Installed" means the object storage area in use and "Configured" means the storage area configured in the schema definition file which maybe still not be applied.

1. only report without applying for all tables.

```
yab database-qadhelp-storage-area-resolve
```

output

```
$ yab database-qadhelp-storage-area-resolve

Table          Object          Type  Installed  Configured  Dump/Load
-----
flhd_det       flhd_det         Tbl   HELP      Schema Area  No
flhk_mstr      flhk_keyword     Idx   HELP_IDX  HELP        No
flhk_mstr      flhk_lang        Idx   HELP_IDX  HELP        No
flhm_mst       flhm_call_pg     Idx   HELP_IDX  Schema Area  No
flhm_mst       flhm_lang        Idx   HELP_IDX  HELP        No
qadhelp_ver    oid_qadhelp_ver  Idx   HELP_IDX  Schema Area  No
xd_mstr        xd_exec          Idx   HELP_IDX  HELP        No
xd_mstr        xd_nbr           Idx   HELP_IDX  HELP        No
xf_mstr        xf_field         Idx   HELP_IDX  Schema Area  No
xf_mstr        xf_pgm           Idx   HELP_IDX  HELP        No

BUILD SUCCESSFUL (47.542 s)
```

2. resolve specified tables storage area inconsistencies.

```
yab database-qadhelp-storage-area-resolve -resolve -tables:flhm_mst,flhd_det
```

output

```
$ yab database-qadhelp-storage-area-resolve -resolve -tables:flhm_mst,flhd_det

Table          Object          Type  Installed  Configured  Dump/Load
-----
flhd_det       flhd_det         Tbl   HELP      Schema Area  No
flhm_mst       flhm_call_pg     Idx   HELP_IDX  Schema Area  No
flhm_mst       flhm_lang        Idx   HELP_IDX  HELP        No

All changes applied.

BUILD SUCCESSFUL (5.455 s)
```

Appendixes: Processing Workflow

Whole process flowchart

Move storage area flowchart

env-info

The *env-info* command displays the Home Server and application URL's configured for the environment.

```
> yab env-info

Client
      QAD Home http://<host>.qad.com:22000/qadhome
      .NET UI  http://<host>.qad.com:22000/qadui
      QAD Web UI https://<host>.qad.com:22011/qad-central

QXtend
      Inbound http://<host>.qad.com:22087/qxi
      Outbound http://<host>.qad.com:22087/qxo

* To print more detailed information use (-more).

BUILD SUCCESSFUL (0.786 s)
```



The *-more* option can be used to display additional information.

```
> yab env-info -more
```

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](#).



The (-command) option restricts the output to commands and the (-setting) option restricts the output to settings. As a convenience, the command-help and config-help commands are available to search either domain exclusively.



Attention

The (-command) and (-setting) option are not implemented before YAB 1.9, so if the YAB client version is lower than 1.9. Those options should still be (-c) for command and (-s) for setting.

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

```
> yab help
```

PROCESSES

clean	Removes temporary files.
config	Prints configuration settings.
env-info	Prints environment information.
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
    yab config [KEY] [KEY] [KEY]...
    KEY
    The key of the setting(s) to print. The meta character '*' is used to match 0 or more
characters.
    If no keys are defined, all configuration settings with a value are printed, and when (-all)
is
    defined, settings without a value are included as well. When using meta characters, quote the
argument to prevent the shell from evaluating the meta character.

    Ex. Print the version of the 'qracore' package.

    yab config "packages.qracore.version"

    Ex. Print the version of all packages.

    yab config "packages.*.version"

    Ex. Print all package settings.

    yab config "packages.*"

    Ex. Print all database instances.

    yab -instances config "db"

OPTIONS

```

Name	Description
-----	-----
all	When true includes settings without a value. Default: false
skip-resolution	When true references are not resolved. Default: false
remove-unresolved	When true unresolved references are removed. Default: false
skip-value	When true values will not be printed. Default: false
instances	When true the "instances" of the key are printed. NOTE: When enabled, meta characters are ignored and the following options are forced (all=true, skip-value=true, trace=false). Default: false
trace setting was resolved.	When true additional information is printed to show how a configuration Default: false
listorder ordered to match the precedence files	Used to control how trace output is ordered. By default, trace output is precedence of configuration files, such that settings defined in higher are printed before settings defined in lower precedence files. When (-listorder) is true the output is ordered by the setting. List order is typically useful when tracing on a setting that contains a
list of items, (group, grouporder annotations)	where the order of the items is mainly a function of meta information and not the precedence of the files in which it is defined. Default: false

```
> yab help appserver.crm.srvrlogginglevel
```

SETTINGS

appserver.srvrlogginglevel	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-deactivate	Perform an index deactivate.
database-qaddb-index-rebuild	Perform an index rebuild.
database-qaddb-jta-enable	Enable the JTA feature to the 'qaddb' database
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-file-update	Generates the default structure for the 'qaddb' database.
database-qaddb-structure-list	Prints the structure of the 'qaddb' database.
database-qaddb-structure-update	Applies new areas to the 'qaddb' database.
database-qaddb-structure-validate	Validates the structure configuration for the 'qaddb' database.
database-qaddb-table-info	Prints table names and (optionally) the number of records they contain.
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.
keepalive                               denyServerASK - disables server initiated
keepalive                               allowServerASK - enables server initiated
development                             denyClientASK - currently unused - for future
development                             allowClientASK - currently unused - for future
...

```

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.
...	

host-update

If an environment is moved from one host to another host, without changing the filesystem layout, *host-update* updates the application configuration files and database / application settings.

1. Update host in build/config/configuration.properties.

```
host=devel.qad.com
```

2. Execute

```
> yab host-update
```

info

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

The main display includes the following columns:

Column	Description								
1	The package name.								
2	The product version.								
3	The location of the package. <table border="1"> <tr> <td>local</td><td>Local software catalog</td></tr> <tr> <td>remote</td><td>Remote software catalog</td></tr> <tr> <td>disabled</td><td>A disabled package.</td></tr> <tr> <td>devel</td><td>An unreleased developer package.</td></tr> </table>	local	Local software catalog	remote	Remote software catalog	disabled	A disabled package.	devel	An unreleased developer package.
local	Local software catalog								
remote	Remote software catalog								
disabled	A disabled package.								
devel	An unreleased developer package.								
4	Will have the value of <i>(new)</i> for all configured packages that have not been applied with a successful update .								

```
> yab info

INSTANCE

/dr01/qadapps/qea

MODULES

|- archiving                1.0.0.78      local
|- assistance-help-ee      2017.0.0.3    local
|- base-api                1.1.0.79      local
|- fin                     2017.0.17.536 local
|- fin-apidoc              2017.1.80.5   local
|- fin-bin64-proxy         2017.1.80.5   local
...

```

Use the *(-depth)* option to show the sub-components of an app.

```
> yab info

INSTANCE

/dr01/qadapps/qea

MODULES

...
|- yab-ee-app              1.6.0.18      local
  |- yab-conv              1.6.0.8        local
  |- yab-core              1.6.0.17      local
    |- dde-ant             2.5.0.1        local
    |- dde-build           2.5.0.122     local
    |- dde-core            2.5.0.4        local
    |- dde-registry        2.5.0.3        local
    |- qpm                 3.0.0.10      local
    |- yab                 1.6.0.56      local
    |- yab-client          1.6.0.56      local
    |- yab-install         1.6.0.35      local
    |- yab-test            1.6.0.1        local
  |- yab-ee-foundation     1.6.0.116     local
  |- yab-qra              1.6.0.52      local
  |- yab-qxtend           1.6.0.48      local

```

The output when a sub-component is overridden with a different version (dde-build-2.5.0.122 => dde-build-2.5.0.122).

```
> yab info

INSTANCE

/dr01/qadapps/qea

MODULES

...
|- yab-ee-app*                1.6.0.18      local
|- yab-conv                  1.6.0.8       local
|- yab-core*                 1.6.0.17      local
|   |- dde-ant               2.5.0.1       local
|   |- dde-build* (2.5.0.122) 2.5.0.123     local (new)
|   |- dde-core              2.5.0.4       local
|   |- dde-registry          2.5.0.3       local
|   |- qpm                   3.0.0.10      local
|   |- yab                   1.6.0.56      local
|   |- yab-client            1.6.0.56      local
|   |- yab-install           1.6.0.35      local
|   |- yab-test              1.6.0.1       local
|- yab-ee-foundation         1.6.0.116     local
|- yab-qra                   1.6.0.52      local
|- yab-qxtend                1.6.0.48      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 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-gracore-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

[build/config/packages/yab-ee-foundation/1/5/0/16/default/yab-ee-foundation.properties]
# @port +0
tomcat.default.httpport=

1.050 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-qadcass-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](#) command.

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 command *system-diagnostics* creates an archive that contains diagnostic files from the environment.

The archive includes the following files and directories:

- All files within the configuration directory (appdir.config).
- All files within the log directory (appdir.logs).
- The output of the info command with the (-more) flag enabled.
- A snapshot.xml includes installed package names and versions.
- A packages.properties file that lists all packages and patches from the environment.
- All files within the customizations directory with (-include-customizations) flag enabled.
- A file listing the structure of the customizations directory with (-include-customizations) flag disabled.
- All files within the patches directory with (-exclude-patches) flag disabled.
- A file listing the structure of the patches directory with (-exclude-patches) flag enabled.

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

[APPDIR DIRECTORY NAME] - [TIMESTAMP].zip

Options

Name

Description

-----	-----
archive	The name of the archive file.
threshold exceeded,	The maximum number of megabytes to capture from each file. If the threshold is the threshold number of megabytes will be captured from the end of the file. Default: 10
include-customizations	Include the files within the customizations directory.
exclude-patches	Exclude the files within the patches directory.

Examples

Ex. Create a diagnostics archive in the current working directory:

```
> yab system-diagnostics
```

EX. Create a diagnostics archive including files within the customizations directory:

```
> yab system-diagnostics -include-customizations
```

Ex. Create a diagnostics archive excluding patches directory:

```
> yab system-diagnostics -exclude-patches
```

Ex. Create a diagnostics archive named myenv.zip:

```
> yab -archive:/dr01/myenv.zip system-diagnostics
```

EX. Create a diagnostics archive named 'myenv.zip' where no log file included in the archive will exceed 5 MB

```
> yab -archive:/dr01/myenv.zip -threshold:5 system-diagnostics
```

system-find

The *system-find* command searches for files referenced from configuration settings.

Ex. Find the data file that distributes the 'ab001' browse

```
> yab find data | xargs grep -l ab001
/dr01/qadapps/qa/build/catalog/packages/mfg/2019/0/19/787/data/qadadm/brw_mstr-brwt_det-brwf_det-
proc_det-prol_det.xml
/dr01/qadapps/qa/build/catalog/packages/mfg/2019/0/19/787/data/qadadm/vue_mstr-vwj_det-vuf_det.
xml
```

Ex. Find the tax programs reached from the operational compile.

```
> yab -includes:"**/us/tx/*.p" find code.mfg
/dr01/qadapps/qa/build/catalog/packages/mfg/2019/0/19/787/src/us/tx/txtx2upd.p
/dr01/qadapps/qa/build/catalog/packages/mfg/2019/0/19/787/src/us/tx/txdetrp3.p
/dr01/qadapps/qa/build/catalog/packages/mfg/2019/0/19/787/src/us/tx/txavconf.p
/dr01/qadapps/qa/build/catalog/packages/mfg/2019/0/19/787/src/us/tx/txcalcIn.p
...
```

system-package-create

The *system-package-create* command creates a package.

Ex. Create a package named 'test' that distributes all of the files in the current working directory.

```
> yab -class:test system-package-create .
> ls -l test-1.0.0.0.zip
-rw-r--r-- 1 wtb devel 804 Dec 15 13:48 test-1.0.0.0.zip
```

Ex. Create another version of the package named 'test' that distributes all of the files in the current working directory.

```
> yab -class:test -version:1.1 system-package-create .
> ls -l test-1.1.0.0.zip
-rw-r--r-- 1 wtb devel 804 Dec 15 13:49 test-1.1.0.0.zip
```

Ex. Create a package while defining some package attributes.

```
> yab -class:another -version:1.2.3.4 -attribute:module.uri=urn:module:com.qad.another -attribute:
owner=abc system-package-create .
> ls -l another-1.2.3.4.zip
-rw-r--r-- 1 wtb devel 804 Dec 15 13:50 another-1.2.3.4.zip
```

Ex. Create a new package by repackaging an existing package with updates from a directory named 'updates'.

```
> yab -copy:test-1.0.0.0.zip -version:1.1 system-package-create updates
> ls -l test-1.1.0.0.zip
-rw-r--r-- 1 wtb devel 804 Dec 15 13:51 test-1.1.0.0.zip
```

Ex. Obtain a version from a version sequence.

```

> cat test.version
<version-specification>
  <version>1.0.0.0</version>
</version-specification>

> yab -class:test -version-sequence:test.version system-package-create

                        system-package-create (1 task)
-----
1/1 system-package-create                                OK (2.260 s)
-----

BUILD SUCCESSFUL (3.296 s)

> ls -l test*
-rw-rw-r-- 1 wtb devel 300 Dec 16 2016 test-1.0.0.0.zip
-rw-rw-r-- 1 wtb devel 79 Dec 16 2016 test.version

> cat test.version
<version-specification>
  <version>1.0.0.1</version>
</version-specification>coli49:sr1-core-systest>

> yab -class:test -version-sequence:test.version system-package-create

                        system-package-create (1 task)
-----
1/1 system-package-create                                OK (0.882 s)
-----

BUILD SUCCESSFUL (1.818 s)

> ls -l test*
-rw-rw-r-- 1 wtb devel 300 Dec 16 06:00 test-1.0.0.0.zip
-rw-rw-r-- 1 wtb devel 299 Dec 16 2016 test-1.0.0.1.zip
-rw-rw-r-- 1 wtb devel 79 Dec 16 2016 test.version

> cat test.version
<version-specification>
  <version>1.0.0.2</version>
</version-specification>

```



The primary value of a package is that it associates a version with a set of files. When it is necessary to distribute updates to these files, a new package should be created with a higher version. Reusing versions eliminates one of the primary benefits of packages.

system-process-list

The *system-process-list* command displays the commands that would be executed if a command were executed.

Ex. Display the commands that would be executed if the environment was updated.

```
> yab system-process-list update

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
...
```

Ex. Display the commands that would be executed if the environment was started.

```
> yab system-process-list start

1. adminserver-start
2. nameserver-start
3. database-qadadm-start
4. database-qaddb-start
5. database-qadhlp-start
6. database-qxevents-start
7. database-qxodb-start
8. database-start
9. appserver-fin-start
10. appserver-mfg-start
...
```

system-updates-list

- [Introduction](#)
 - [Options](#)
 - [Usage & Examples](#)
 - [Default](#)
 - [-package](#)
 - [Usage](#)
 - [Example](#)
 - [-version-level](#)
 - [Usage](#)
 - [Example](#)
 - [-skip-compliance](#)
 - [Usage](#)
 - [Usage](#)
 - [-all](#)
 - [Usage](#)
 - [-depth](#)
 - [Usage](#)
 - [Example](#)
-

Introduction

The command *system-updates-list* can provide user a convenient approach to check for updates to modules installed in the YAB instance.

Options

```
> yab help system-updates-list
```

PROCESS

system-updates-list - Lists the updates that are available.

OPTIONS

Name	Description
package	Lists updates for a package by name (e.g. 'abc'). Wildcard matching is supported. *: 0 or any multiple character(s) .: any single character e.g. foo* => foo, foo-bar, ... fo. => foo, fox, fog, ... *-bar => any string ends with "-bar" *foo* => any string contains "foo"
version-level	Used to control what types of updates are listed. Possible values: major, minor, patch, build. Default: patch A version consists of four parts: e.g. 1.2.3.4 Major = 1 Minor = 2 Patch = 3 Build = 4 When the version-level is set to 'patch', listing updates in an environment with the package 'abc-1.2.3.4' configured would display the highest 'abc-1.2.x' version available in all of the configured catalogs. Likewise with a version-level set to 'minor', the highest 'abc-1.x' version.
skip-compliance	List updates for all packages and not only the packages that conform to QAD versioning standards.
exclude-current-version	Do not show the current version of packages in the display. Default: Show the current version column.
all	Include packages that are up to date. Default: Only output the packages which have updates available.
depth	Used to limit which modules that are printed. A depth of 0 prints all modules, whereas a depth of 1 will only print the top level modules, members and so on. a depth of 2 will print the top level modules and their direct members and so on. Default: 1. e.g. -depth:2

Usage & Examples

Default

Option	Default value
package	-
version-level	patch
skip-compliance	false
exclude-current-version	false
all	false
depth	1

By default, the command will only list packages which follow the versioning standard and have available patch updates.

Only the top level packages will be displayed in the output.

```
> yab system-updates-list
```

-package

The *-package* option limits the check for updates to the specified package (s). Wildcard matching is supported to include multiple packages.

Usage

```
yab system-updates-list -package:name|pattern
```



Wildcard matching

Here we ONLY support two wildcard '.' and '*'

- * is for 0 or any multiple character(s)
- . is for any single character

Example

1.Specify a single package

```
> yab system-updates-list -package:foo-app1
```

2.Specify all the package(s) which contains "app"

```
> yab system-updates-list -package:*app*
```

-version-level

The option *-version-level* is to restrict the output list in a specified version level.

Usage

```
yab system-updates-list -version-level:major|minor|patch|build
```



Versioning convention

A version consists of a set of four digits arranged from the most significant to the least significant value (when read from left to right).

The integer values are associated with the following names: major, minor, patch, build.

e.g. foo-bar-1.2.3.4 (Package name is foo-bar, major = 1, minor = 2, patch = 3, build = 4.)

Example

Check for any updates up to and including minor version changes.

```
> yab system-updates-list -version-level:minor
```

-skip-compliance

The option *-skip-compliance* is used to check updates for all packages regardless of whether they conform to version standards.

Usage

```
> yab system-updates-list -skip-compliance

INSTANCE

/qad/sbox/007/user/ucm/02/trunkcore

CATALOGS

/qad/local/sandbox/cache
http://packages.qad.com
http://plli05:32000/

MODULES                                CURRENT                AVAILABLE              UP-TO-DATE

|- base-api                           1.1.0.79               1.1.2.3                No
|- dde-build                           2.5.0.97               2.5.0.109              No
|- fin-apidoc                          2017.0.80.4            2017.0.80.7            No
|- fin-bin64-proxy                     2017.0.80.4            2017.0.80.7            No
|- fin-bin64-qadfin                    2017.0.80.4            2017.0.80.7            No
|- fin-bldata                          2017.0.80.4            2017.0.80.7            No
|- fin-customization                    2017.0.80.4            2017.0.80.7            No
|- fin-erp-trunk                       2017.0.17.497          2017.0.17.504          No
|- fin-src-proxy                       2017.0.80.4            2017.0.80.7            No
|- fincore                             2.17.0.8               2.17.1.1               No
|- foo-pkg2                            1.3.0.0                1.3.1.1                No
|- foo-pkg3                            3.3.0.0                3.3.1.1                No
|- mfg-erp-trunk                       2017.0.17.497          2017.0.17.504          No
|- mfgcoreplus                         2.17.0.15              2.17.1.3               No
|- mfgcoreplus-api                    2.17.0.15              2.17.1.3               No
|- netui                               3.2.0.62               3.2.1.12               No
|   netui-module-32-cumulative          3.2.38.0               3.2.55.0               No
|- netui-module-cmr                    1.8.0.33               1.8.0.39               No
|- netui-module-ctd                     6.0.0.3                6.0.0.4                No
|- netui-module-fin                    2017.0.80.5            2017.0.80.9            No
|- netui-module-grs                     3.0.0.3                3.0.2.8                No
|- netui-module-kanban                 1.2.3.10               1.2.3.11               No
|- netui-module-mrc                     1.3.0.82               1.3.0.85               No
|- netui-module-mswpsw                 3.3.8.7                3.3.7000.35            No
|- netui-module-periodic-costing        1.0.0.200              1.0.0.205              No
|- netui-module-wms                     1.0.0.1                1.0.0.20               No
|- qpm                                 3.0.0.5                3.0.0.8                No
|- qracore                             2.17.0.32              2.17.5.0               No
|- qracore-api                         2.17.0.32              2.17.5.0               No
|- qxtend-bundled                       1.8.7.11               1.8.8.3                No
|- requisition-api                     1.1.0.91               1.1.2.0                No
|- system-test-data-erp-trunk           2017.0.17.497          2017.0.17.504          No
|- tomcat                              7.0.52.0               7.0.77.1               No
|- trans-dotdees-fhd                    9.3.16.1               9.3.16.8               No
|- yab                                 1.6.0.30               1.6.0.46               No
|- yab-client                           1.6.0.21               1.6.0.42               No
|- yab-conv                             1.6.0.5                1.6.0.6                No
|- yab-conv-ee-xrc                     2016.0.0.1             2016.0.5.0             No
|- yab-ee-foundation                    1.6.0.89               1.6.0.108              No
|- yab-install                          1.6.0.24               1.6.0.28               No
|- yab-qla                              1.6.0.34               1.6.0.44               No
|- yab-qxtend                           1.6.0.44               1.6.0.45               No

...
```

-exclude-current-version

The option *-exclude-current-version* is to hide the current version column in the output list. The current version column will be displayed by default.

Usage

```
> yab system-updates-list -exclude-current-version
```

-all

This option **-all** is a row control option. The command will only output the package(s) which have updates available by default. By using this option the output will list all the packages whether or not they have updates available.

Usage

```
> yab system-updates-list -all
```

-depth

This option is used to limit the modules which are displayed. The command will output all package(s) include all the inner package(s) no matter how deep it is by using value 0 in this option.

By default, the command will only list the top level packages.

Usage

```
yab system-updates-list -depth:n
```

Example

1. List all the package(s) no matter how deep it is.



Asterisk in up-to-date column

In default case, the process will just show the package(s) which have updates available. That means only the package that column UP-TO-DATE is "No" will be displayed.

If you see an asterisk on the right of "Yes", that means this app contains some packages which have updates available in current condition but hidden by the depth control. You can use option **-depth** to spread it.

```

> yab system-updates-list -depth:0

INSTANCE

/qad/sbox/007/user/ucm/02/trunkcore

CATALOGS

/qad/local/sandbox/cache
http://packages.qad.com
http://plli05:32000/

MODULES                CURRENT                AVAILABLE                UP-TO-DATE

|- base-api             1.1.0.79              1.1.2.3                  No
|- foo-app1             1.0.0.0               1.0.0.0                  Yes*
  |- foo-app-lvl2       1.0.0.0               1.0.1.0                  No
    |- foo-m1           2.0.0.0               2.0.0.2                  No
    |- foo-m3           1.3.0.1               1.3.1.0                  No
    |- foo-test         1.0.0.0               1.0.1.1                  No
|- foo-pkg1             1.1.0.0               1.1.1.1                  No
|- foo-pkg3             3.3.1.0               3.3.1.1                  No
|- mfgcoreplus          3.0.0.1               3.0.0.2                  No
|- mfgcoreplus-api      3.0.0.1               3.0.0.2                  No
|- qracore              3.0.0.2               3.0.0.11                 No
|- qracore-api          3.0.0.2               3.0.0.11                 No
|- requisition-api      1.1.0.91              1.1.2.0                  No
|- yab-conv             1.6.0.5               1.6.0.6                  No
|- yab-conv-ee-xrc      2016.0.0.1            2016.0.5.0               No
|- yab-core             1.6.0.0               1.6.0.4                  No
  |- dde-build          2.5.0.105             2.5.0.109                No
  |- yab-client         1.6.0.38              1.6.0.42                 No
  |- yab-install        1.6.0.25              1.6.0.28                 No
  |- yab                1.6.0.39              1.6.0.46                 No
|- yab-ee-foundation    1.6.0.103             1.6.0.108                No
|- yab-gra              1.6.0.41              1.6.0.44                 No
|- yab-qxtend           1.6.0.44              1.6.0.45                 No

...

```

2.List specified depth package(s)

```
> yab system-updates-list -depth:2
```

INSTANCE

/qad/sbox/007/user/ucm/02/trunkcore

CATALOGS

/qad/local/sandbox/cache
<http://packages.qad.com>
<http://plli05:32000/>

MODULES	CURRENT	AVAILABLE	UP-TO-DATE
- base-api	1.1.0.79	1.1.2.3	No
- foo-app1	1.0.0.0	1.0.0.0	Yes*
- foo-app-lvl2	1.0.0.0	1.0.1.0	No
- foo-test	1.0.0.0	1.0.1.1	No
- foo-pkg1	1.1.0.0	1.1.1.1	No
- foo-pkg3	3.3.1.0	3.3.1.1	No
- mfgcoreplus	3.0.0.1	3.0.0.2	No
- mfgcoreplus-api	3.0.0.1	3.0.0.2	No
- qracore	3.0.0.2	3.0.0.11	No
- qracore-api	3.0.0.2	3.0.0.11	No
- requisition-api	1.1.0.91	1.1.2.0	No
- yab-conv	1.6.0.5	1.6.0.6	No
- yab-conv-ee-xrc	2016.0.0.1	2016.0.5.0	No
- yab-core	1.6.0.0	1.6.0.4	No
- dde-build	2.5.0.105	2.5.0.109	No
- yab-client	1.6.0.38	1.6.0.42	No
- yab-install	1.6.0.25	1.6.0.28	No
- yab	1.6.0.39	1.6.0.46	No
- yab-ee-foundation	1.6.0.103	1.6.0.108	No
- yab-qla	1.6.0.41	1.6.0.44	No
- yab-qxtend	1.6.0.44	1.6.0.45	No

...

update

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

```
> yab update
```



The update command will restart the environment. Some configuration changes can be applied with the [reconfigure](#) command which does not take down the environment.

validate

The *validate* command is used primarily to validate YAB configuration settings. The command, for example, will check that all required settings have a value, that values that expect an integer are assigned a valid value, and so on. A subset of these validations are automatically performed every time the environment is [refreshed](#).

See [Validation Settings](#)

System Configuration

System Configuration includes topics related to (re)configuring an environment.

The standard configuration procedure (discussed briefly in the [introduction](#)) is to define the configuration changes in *build/config/configuration.properties* and then apply the change to the environment with the appropriate YAB command. The [update](#) command may be used to apply all changes (unless specifically stated otherwise) but it is comprehensive and will take the environment offline for a period of time. In certain cases, you may receive guidance to apply the change with a different command (usually [reconfigure](#)) as a more efficient alternative.

Example Reconfiguration

We can demonstrate this process flow in action by enabling and disabling 4GL Tracing (extra logging) on the MFG application server.

List the application servers in the configuration to find/remember the name of the MFG application server in the configuration.

```
> yab -instances config appserver
appserver.fin
appserver.mfg
appserver.qra
...
```

The settings associated with an application server correspond very closely to the application server settings in the Progress configuration file *ubroker.properties*. Print the help for the application server.

```
> yab help "appserver.mfg."
```

Using that information, enable 4GL tracing, by editing *configuration.properties* and adding the following settings to override the default values (in all likelihood *configuration.properties* will not already contain these settings, but if it did we would edit the existing settings in the file).

build/config/configuration.properties

```
appserver.mfg.svrlogginglevel=4
appserver.mfg.svrlogentrytypes=ASPlumbing,DB.Connects,4GLTrace
```

Execute the `appserver-mfg-update` command to apply the configuration change to the environment (which calls the Progress *mergeprop* utility to merge the MFG application server configuration into *build/work/generated/ubroker.properties*). This command is included in a [reconfigure](#) and everything in a [reconfigure](#) is included in an [update](#), so we also could use either of these commands to apply the change. A good rule of thumb is that you can use [reconfigure](#) for updating scripts and changing most Progress settings, but many other changes will require an [update](#).

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

The changes to *ubroker.properties* will be picked up by the application server without restarting because we have enabled runtime updates (other Progress settings may not support dynamic updates).

```
> yab config appserver.mfg.allowruntimeupdates
appserver.mfg.allowruntimeupdates=1
```

With 4GL Tracing enabled, the application server will write a lot more information to its log files and this can slow down the application and consume excessive amounts of disk space, so in this special case it is a good idea to undo the configuration change after we have captured the data to evaluate. To undo the change, remove the settings from *configuration.properties* and then apply the changes using the same command as before.

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

General Information

Configuration Type System

Many of the configuration settings are related by the fact that they describe the desired configuration of a specific resource in the environment. In cases where there is only one instance of the resource in the environment (and there will only ever be one instance) we use the following pattern to configure the resource:

[TYPE] . [NAME]

The Progress Admin Server and Progress Name Server are examples of this type of resource.

Ex. Admin Server

```
> yab config "adminserver.*"
adminserver.adminport=22082
adminserver.conmgr=/qad/sandbox/user/wtb/02/sc2-core/build/work/generated/conmgr.properties
adminserver.plugins=/qad/sandbox/user/wtb/02/sc2-core/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.
allowArraySyntax=true -Dserver.start.retryCount=10 -Dserver.start.retryInterval=3
adminserver.port=22001
adminserver.ubroker=/qad/sandbox/user/wtb/02/sc2-core/build/work/generated/ubroker.properties
```

To describe resources that may have multiple instances in the environment we use the following pattern:

[TYPE] . [INSTANCE] . [NAME]

A Progress database is an example of this type of resource.

Ex. QADDB database

```
> yab config "db.qaddb.*"
db.qaddb.bi=16384
db.qaddb.biblocksize=16
db.qaddb.blocksize=8
db.qaddb.centuryformat=1950
db.qaddb.codepage=utf-8
db.qaddb.collation=ICU-UCA
...
```

Some of these multiple instance resources are setup to inherit settings from a prototype instance named `_base`.

Ex. Prototype database configuration

```
> yab config "db._base.*"
db._base.bi=16384
db._base.biblocksize=16
db._base.blocksize=8
db._base.centuryformat=1950
db._base.codepage=utf-8
db._base.collation=ICU-UCA
...
```

A setting will be inherited from the prototype unless specifically overridden. The `(-trace)` option of the `config` command shows this relationship.

```
> yab -trace config db.qaddb.codepage
db.qaddb.codepage=utf-8 (inherited: db._base)

[build/config/packages/yab-ee-foundation/1/5/0/16/default/yab-ee-foundation.properties]
db._base.codepage=utf-8
```

Here is a table listing some of the important configuration types in the system.

Type	Description	Multiple Instances
adminserver	Progress Admin Server	NO
aia	Application Internet Adapter	YES

Proprietary of QAD, Inc.

appserver	Application Server	YES
code	Code	YES
bootstrap	QRA Bootstrap Documents	YES
copy	File copy	YES
data	Data	YES
db	Database	YES
dbserver	Database Server	YES
filegen	File merge and replace	YES
netui	.NET configuration	NO
ns	Name Server	NO
packages	Packages	YES
process	Commands	YES
qpackage	Client packages	YES
schema	Schema	YES
tomcat	Tomcat	YES
webapp	Web Application	YES
ws	WebSpeed Server	YES
wsa	Web Services Adapter	YES

Configuration Files

- [Comments](#)
- [References](#)
- [Annotations](#)
 - [@if](#)
 - [@unless](#)
 - [@end](#)
 - [Built-in If/Unless Tests](#)
 - [@extends](#)
 - [@aliasof](#)
 - [@ref](#)
 - [@append](#)
 - [@noappend](#)
 - [@group](#)
 - [@grouporder](#)
 - [@port](#)
 - [@exclude](#)
 - [@patch](#)
 - [@unique](#)
 - [@multiply](#)
- [Indexers](#)
 - [Examples](#)

Configuration settings are defined in configuration files where each setting is defined on a separate line using the syntax:

```
KEY=VALUE
```

By convention keys are lowercase and the period is used as a delimiter for compound names (e.g. `dbserver.qaddb.afterimagebuffers`). The [type system](#) describes additional conventions for configuring "resources" in the environment.

Comments

A line that starts with the hash (#) character is processed as a comment.

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

References

The value of a setting may reference other settings using the syntax:

```
${[REFERENCE]}
```

Ex. Configure the server log file for the MFG application server.

```
appserver.mfg.svrlogfile=${appdir.logs}/${name}.server.log
```

A reference is resolved using the following algorithm:

- Resolve the reference relative to the "ancestors" of the referencing key (closest match wins).
- Resolve the reference as an absolute reference.
- Resolve the reference as a Java system property.

Ex. Hello World

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

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

References can be nested.

In the following example the setting `russian.dolls` has the value "d".

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

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

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

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

```
${<[REFERENCE]}
```

In the following example the setting "d" has the value "c" because "a" is aliased to "c" and "d" dereferences "a". The setting "e" has the value "b" because the setting "b" is not aliased to another setting.

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

Wildcards may be used to reference multiple values evaluated to a comma-separated list. In the following example, the value of "another" is "a,b".

```
foo.bar.setting=a
foo.baz.setting=b
another=${foo.*.setting}
```

Annotations

An annotation is a special form of comment that provides additional information about the setting that follows the annotation. Annotations are defined using the syntax:

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



Care must be taken when editing a file that uses annotations. Annotations are formatted as comments but they do have an effect on the system. In particular you should be careful commenting out settings with annotations.

The following file defines the setting "b" with the value "bar" and the setting "a" with the value "foo" if the setting "someprop" is defined.

```
# @if someprop
a=foo

b=bar
```

To temporarily remove the setting "a" from the system the file should be backed up. Commenting out "a" would have the unintended consequence of associating the "@if" annotation with the next setting which in this case is "b".

Ex. Wrong

```
# @if someprop
# a=foo

b=bar
```

@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. The operators '&&' and '||' can be used to evaluate multiple tests with a single IF statement using a Boolean AND/OR respectively to combine the outcomes, where tests are processed from left to right. The outcome from tests defined in separate IF/UNLESS statements are combined using an implicit AND operator.

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 [TEST] [&& | ||] [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. The operators '&&' and '||' can be used to evaluate multiple tests with a single UNLESS statement using a Boolean AND/OR respectively to combine the outcomes, where tests are processed from left to right. The outcome from tests defined in separate IF/UNLESS statements are combined using an implicit AND operator.

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 [TEST] [&& | ||] [TEST]...
# @unless FINAL [TEST]
```

@end

Ends the scope of one or more tests.

Built-in 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.
KEY : ALIAS	foo : ALIAS	Tests if the key is aliased or not.
PATH : FILE	/foo/bar : file	Tests if the file path exists.
PATH : DIR	/foo/bar /dir/ : dir	Tests if the directory path exists.



Care should be taken when nesting if/unless tests. The scope of all tests is ended by the "@end" annotation. The following configuration probably does not do what the author intended. The setting "c" will always be defined (regardless of whether or not "someprop" is defined, because both tests are ended by the first "@end" annotation.

```
@if someprop
a=foo
@if otherprop
b=bar
@end
c=baz
@end
```

@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 the child properties in B are merged to A and the child properties in A are merged to B, where a child property that was defined both in A and B obtains the value from 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
```

An alias setting can be overridden like any other configuration setting. The value that you use to "break" the alias ("_unlink_" in the example below) is not important.

Ex. The QAD Reference Architecture (QRA) database requirements are aliased to the main QADDB database.

```
> yab config db.qadmodule
db.qadmodule=>db.qaddb
```

To deploy QRA schema and data into a separate database.

```
db.qadmodule=_unlink_
```

@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 (otherwise the value 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.qadcp1,db.qadadm,db.qadh1p,db.qadrcode
```

@group

Used in conjunction with @append to obtain a desired value ordering.

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. When the @grouporder annotation is defined for a group it overrides all ordering hints for that group.

An ordering hint is defined using the syntax *[+/-][ID PATTERN]* where the *ID PATTERN* is used to select other group members by their @id and (-) is interpreted as an instruction to order value before the selected values, whereas (+) is interpreted as an instruction to order the value after the selected values.

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=Extensions
Undefined=Default¹

¹ The default group is the group assigned when a group index is not defined. To allow for the possibility of introducing additional non-default groups in the future, members of the default group should not explicitly define a group index.

ID Patterns

An ID pattern may use the '*' character to represent one or more characters and the '?' character to represent a single character. If an ID is matched by multiple patterns, the longest pattern is assigned the match. The '*' character alone may be used to match all ID's that are not otherwise matched by a pattern (and this latter use typically only makes sense in the context of a @grouporder annotation).

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. When the @grouporder annotation is defined for a group it overrides all ordering hints for that group.

Variants

Orders the members of the default (lowest precedence) group:

```
# @grouporder [ID PATTERN] [ID PATTERN] . . .
```

Orders the members of a specific group:

```
# @grouporder [INTEGER] [ID PATTERN] [ID PATTERN] . . .
```

See @group for a discussion of ID patterns.

@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.

Patching occurs before property references are evaluated. Use the (-skip-resolution) option on the `config` command to see the value of the property to be patched before references are evaluated.

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. Merge "-Bt 3000" into the `progress.startup.params` setting replacing an existing value (matched by the regex), otherwise appending the value using a space as a delimiter.

```
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
```

Ex. Remove a PROPATH entry from a Progress session.

```
> yab -skip-resolution config appserver.mfg.propath
appserver.mfg.propath=${appdir}/config,${packages.qracore.dir}/qad.qra.core/bin/qracore.pl,${code.
mfg-customizations.dir},${dist.dir}/fin/proxypatch,${code.ict.dir},${netui.pro.dir}/com
/mfgpro,${packages.fin-bin64-proxy.dir}/proxy.pl,${netui.pro.dir},.,${packages.qra-oe11.dir}/lib
/qra.pl,${code.mfg.dir},${code.mfg.dir}/us,${code.mfg.dir}/us/bbi,${code.activity-feed.
dir},${code.qxi.dir}
```

```
# @patch /,\\${code.ict.dir}/
appserver.mfg.propath=
```

```
> yab -skip-resolution config appserver.mfg.propath
appserver.mfg.propath=${appdir}/config,${packages.qracore.dir}/qad.qra.core/bin/qracore.pl,${code.
mfg-customizations.dir},${dist.dir}/fin/proxypatch,${netui.pro.dir}/com/mfgpro,${packages.fin-
bin64-proxy.dir}/proxy.pl,${netui.pro.dir},.,${packages.qra-oe11.dir}/lib/qra.pl,${code.mfg.
dir},${code.mfg.dir}/us,${code.mfg.dir}/us/bbi,${code.activity-feed.dir},${code.qxi.dir}
```

@unique

Used to eliminate duplicate values associated with a setting. When this annotation is defined, only the highest precedence duplicated value is retained. This is typically used in conjunction with appended values.

@multiply

Multiplies the value of the property.

Variants

```
# @multiply [MULTIPLIER]
Multiplies the property by the multiplier value.
```

```
$ yab config -trace foobar
foobar=5000
```

```
[build/config/configuration.properties]
# @multiply 1000
foobar=5
```

```
# @multiply [MULTIPLIER] [PROPERTY DEFAULT]
```

Multiplies the property by the multiplier value. If the property value is undefined, uses the property default as the property value instead.

```
$ yab config -trace foobar
foobar=10000
```

```
[build/config/configuration.properties]
# @multiply 1000 10
foobar=${baz}
```

Indexers

A configuration indexer is a setting that is used to change the value of another setting and has the following syntax:

KEY[INDEX]=VALUE

Where:

Term	Description								
KEY	The key of the setting to change. The key may use the '*' character to represent any name in a compound KEY. For example, the KEY "appserver*.srvrstartupparam" would update the "srvrstartupparam" of all application servers.								
INDEX	The index is used to match the setting value. The index may have a trailing '*' character to represent 0..n characters. When a trailing '*' is used, the VALUE is required and the value that is applied is the concatenation of the matched prefix and the VALUE. For example, if the INDEX is "foo=" and the VALUE is "baz" and the value of the KEY is "foo=bar", the value of the KEY will be changed to "foo=baz". A character is escaped using a character reference "&#nnnn;" where "nnnn" is the Unicode code point of the character to escape. In practice, the following characters are the only characters that may need to be escaped: <table> <tr> <th>Character</th><th>Character Reference</th></tr> <tr> <td>*</td><td>&#42</td></tr> <tr> <td>=</td><td>&#61</td></tr> <tr> <td>[</td><td>&#91</td></tr> </table>	Character	Character Reference	*	*	=	=	[	[
Character	Character Reference								
*	*								
=	=								
[	[								
VALUE	The value to apply. The following values are given a special interpretation. <table> <tr> <th>Value</th><th>Interpretation</th></tr> <tr> <td>NULL</td><td>Remove single values matched by the INDEX.</td></tr> <tr> <td>NULL_PAIR</td><td>Remove paired values matched by the INDEX.</td></tr> </table>	Value	Interpretation	NULL	Remove single values matched by the INDEX.	NULL_PAIR	Remove paired values matched by the INDEX.		
Value	Interpretation								
NULL	Remove single values matched by the INDEX.								
NULL_PAIR	Remove paired values matched by the INDEX.								



A primary use case of indexers is to update settings that are argument lists. In some cases, the system can infer whether an argument is a unary argument or is paired with another argument. For example, the Progress startup parameter (-Bt) expects the next argument will configure the number of buffers, whereas (-b) is a unary argument that stands on its own. When removing arguments the NULL and NULL_PAIR values provide the system with the information it needs to do the right thing in all circumstances. When adding/updating unary arguments that contain both a key and a value without using a standard delimiter like "=" or ":", it is best to use wildcards in the INDEX to select the argument, which allows the system to do the right thing in all circumstances.

Ex. Updating Java Max Heap Size

```
tomcat.default.startuptopts[-Xmx*]=2048m
```

Indexers are used to update a part of a value without replacing the entire value, which is also how the "@patch" annotation is used. The main difference between indexers and patches is that patches are applied before property references have been evaluated whereas indexers are applied after property references have been evaluated. Patches are able to leverage the expressive power of regular expressions, but in many cases an indexer will provide a more concise, lucid syntax to configure a change.



Using indexers on settings that are referenced by other settings, can lead to non intuitive results, because indexers are applied after property references are evaluated. For example, the setting "progress.startup.params" is a list of Progress startup params used in various Progress sessions. The setting "progress.startup.params.streams" references the "progress.startup.params" setting adding in stream configuration parameters.

```
progress.startup.params.streams=${progress.startup.params} -cpinternal ${db._base.codepage} ...
```

If "progress.startup.params" is updated using an indexer it will not be reflected in "progress.startup.params.streams" because the reference is evaluated before the indexer is applied.

The solution is to use the "@patch" annotation or apply the indexer to the settings that reference the setting (i.e. "progress.startup.params.streams" in this example).

The technique used to apply an indexer to a setting depends on the strategy associated with the setting:

Strategy	Description
list	Interpret the value as a list.
args	Interpret the value as a list of arguments.

The default strategy is to process a setting as a comma delimited list. The standard settings are associated with an appropriate indexer strategy. The "@indexer" annotation may be used to configure the strategy for custom settings.

```
# Support indexers that process the value as a whitespace delimited list of arguments.
# @indexer args
custom.setting=

# Support indexers that process the value as a colon separated list.
# @indexer list :
custom.setting=
```

Examples

Ex. Set the value of the parameter (-Bt) to 20000 adding the value if it does not exist, otherwise replacing the existing value.

```
progress.startup.params[-Bt]=20000
```

Ex. Remove the parameter (-Bt) and its value if they exist.

```
progress.startup.params[-Bt]=NULL
```

Ex. Add the parameter (-v6colon) if it does not exist.

```
progress.startup.params[-v6colon]=
```

Ex. Remove the parameter (-v6colon) if it exists.

```
progress.startup.params[-v6colon]=NULL
```

Ex. Change the "-Xms" value of all Tomcat instances.

```
tomcat.*.startuptopts[-Xms*]=512m
```

Ex. Add a database to the operational compile.

```
code.mfg.databases[db.custom]=
```

Ex. Remove a database from the operational compile.

```
code.mfg.databases[db.custom]=NULL
```

Distributing Settings

A convenient way to distribute a group of configuration settings is to define the settings in a file that can be copied into (and out of) an environment.

The `(-p)` option of the [YAB Client](#) can be used to bring configuration settings into an environment.

```
yab -p: [FILE|URL]
```

The client will process any [include](#) statements in the referenced document and then copy the document into the *build/config/system* directory. If a file of the same name exists in the *build/config/system* directory it will be replaced. YAB will automatically adjust the configuration but the changed configuration would still need to be applied to the environment as described in the [introduction](#).

Alternatively, when dealing with configuration documents that do not use include statements, the files can be directly copied into the *build/config/system* directory using the OS file copy command.

```
> cp /shared/configs/development.properties build/config/system
```

Use the OS file remove command to remove the settings from the environment (and apply the change to the system as described in the [introduction](#)).

```
> rm build/config/system/development.properties
```

Reference the [config.order](#) setting to adjust the precedence order of system documents.

Merge Settings

The `-merge-settings` option of the [YAB Client](#) is used to merge the configuration documents referenced by the `(-p)` option into a consolidated configuration document, rather than bring the documents into the active environment.

`-merge-settings:FILE`

The `-excludes` option may be used to exclude a list of include files from the result.

`-excludes:NAME1,NAME2...`

Ex.

```
yab -p:URL -p:URL -excludes:NAME -merge-settings:FILE
```

See: [Configuration File Includes](#)

Configuration File Includes

The **YAB Client** supports evaluating include statements in configuration files when they are brought into the environment using the **(-p)** option. In the environment the include statements are replaced by the referenced document.

Include statements are defined using the following 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:

DIR EC TIVE	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. <pre>yab -p:/opt/media/ee2016.properties -DEVEL -PROD -v create</pre>
URL	A well formed URL (e.g. "http://yab.qad.com/core.properties"). Query string parameters associated with the requested document will be used when requesting included documents. For example if the following request was submitted to the client: <pre>yab -p:"http://yab.qad.com/core.properties?rev=920" -v create</pre> And if the core-systest.properties document had the following definition: core-systest.properties <pre>{a.properties} {b.properties}</pre> The client would retrieve the includes using the following URLs: <pre>http://yab.qad.com/a.properties?rev=920 http://yab.qad.com/b.properties?rev=920</pre>
AB SO LU TE PA TH	An absolute path to a file (e.g. "/dr01/configurations/core.properties").
RE LA TIV E PA TH	A relative path (e.g. "configurations/core.properties"). The path is resolved relative to the document defining the include statement. The notation "../" is used to signify a parent path.

Undefined Progress Parameters

Most Progress parameters can be configured through a corresponding YAB setting. For example, to list all of the YAB configuration settings for an application server use the [help](#) command:

```
> yab config-help appserver
```

When a valid Progress parameter does not have a corresponding YAB setting, the Progress parameter may still be configured using the following syntax:

Parameter (without value)

```
[aia|appserver|dbserver|ns|webspeed|wsa].[INSTANCE].property.[PROPERTY INSTANCE].key
```

Ex. Define a new Progress parameter named "ParameterWithoutValue" that does not have a value on the QADDB (e.g. - ParameterWithoutValue).

```
dbserver.qaddb.property.pwov.key=ParameterWithoutValue
```

NOTE: The PROPERTY INSTANCE ("pwov" in the preceding example) is a name that uniquely identifies the setting and can be chosen at your discretion.

Parameter (with value)

```
[aia|appserver|dbserver|ns|webspeed|wsa].[INSTANCE].property.[PROPERTY INSTANCE].key
```

```
[aia|appserver|dbserver|ns|webspeed|wsa].[INSTANCE].property.[PROPERTY INSTANCE].value
```

Ex. Define a new Progress parameter named "ParameterWithValue" with the value "512" on the QADDB (e.g. - ParameterWithValue 512).

```
dbserver.qaddb.property.pwv.key=ParameterWithValue  
dbserver.qaddb.property.pwv.value=512
```

Reconfiguring Databases

Databases are associated with the `db` and `dbserver` configuration types.

The `db` type configures the database.

`db . INSTANCE . *`

Ex.

`db . qaddd . *`

And the `dbserver` type configures the database server (for databases with a database server)

`dbserver . INSTANCE . *`

Ex.

`dbserver . qaddd . *`

The database and server are associated by sharing the same instance name (e.g. "qaddd").

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 installer.) To change the location of an existing database, the database must either be rebuilt using the command `db-[INSTANCE]-rebuild` (which will only retain system data) or repaired using the Progress *prstrt* 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 the areas and extents of the database. For detailed information about structure description files, see *OpenEdge Data Management: Database Management*.

The following command will display a representation of the current structure of a database

```
database-[INSTANCE]-structure-list
```

It will also write the current structure of the database to a file in the database directory:

```
[PHYSICAL NAME].st
```

The structure file in the database directory should always reflect the current structure of the database to support disaster recovery.

To customize the database structure configure the *structurefile* setting on the database.

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



The database structure should be configured before the environment is created (see QAD Enterprise Edition Installation Guide for details on providing install properties to the installer.) To change the structure of an existing database, the database must either be rebuilt using the command *db-[INSTANCE]-rebuild* (which will only retain system data) or adjusted using the Progress *prstrct* tool.

Configuring 4GL Broker Network Support

To enable the 4GL broker to service connections over a TCP/IP network enable the broker's *network* setting which will assign the broker a port from the configured [port range](#).

Ex. Configure the QADDB database to accept connections over the network with an auto-assigned port.

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

Alternatively you can assign the broker port explicitly.

Ex. Configure the QADDB database to accept connections over the network with an explicit port.

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

Use the `database-script` command to regenerate the database start scripts.

```
> yab database-script
```

Use the `reconfigure` command to apply this change.

```
> yab reconfigure
```



The core application components will use shared memory connections (for performance reasons) even when the 4GL broker is configured to allow network connections. The setting "progress.database.network" can override this behavior but it should not be changed.

Configuring a SQL-92 Secondary Login Broker

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

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

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

Alternatively you can assign the broker port explicitly.

Ex. Configure the QADDB database to accept SQL connections over the network with an explicit port.

```
dbserver.qaddb.sql.port=23500
```

Use the `database-script` command to regenerate the database start scripts.

```
> yab database-script
```

Configuring Before Imaging

To enable database before-imaging on a database:

```
dbserver.INSTANCE.beforeimageprocess=true
```

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

Use the `database-script` command to regenerate the database start scripts.

```
> yab database-script
```

Configuring After Imaging

Overview

After-imaging 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 after-image (AI) files. You can use the AI files in the roll-forward recovery process to restore 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

Update Database Structure

The QAD default database structure does not define after image extents.

Use the following command to refresh the exported database structure.

```
database-[INSTANCE]-structure-list
```

And then review the refreshed database structure file to determine if the database has AI extents defined (search for lines that begin with an "a").

```
databases/[PHYSICAL NAME].st
```

If the database does not have any AI extents they will need to be added. Review the Progress documentation for details on properly sizing AI extents ("BASIC GUIDE TO AFTER-IMAGING"). The following example is only for demonstration.

Create a temporary file that defines the AI extents to add.

Ex. admdb-ai.st

```
a /dr01/qadapps/qea/databases/admdb.a1
a /dr01/qadapps/qea/databases/admdb.a2
```

Add the AI extents to the database using the Progress *prostrct* command and then refresh the exported database structure file.

```
> . scripts/pset
> prostrct add databases/admdb admdb-ai.st
> prostrct list databases/admdb
```

Enable 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
```

```
dbserver.qadadm.afterimageprocess=true
```

Use the `database-script` command to regenerate the database start scripts.

```
> yab database-script
```

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
```

Recovery

Consult the Progress documentation for instructions on roll-forward recovery.

Configuring Codepage and Collation

The default and recommended configuration of Enterprise Edition creates databases using the UTF-8 codepage and ICU-UCA collation.

To configure the codepage and/or collation for all databases adjust settings on the `db._base` configuration instance which will be inherited by all databases.

Ex. Configure ISO8859-1 codepage and Spanish collation.

```
db._base.codepage=ISO8859-1
db._base.template=spa
db._base.collation=SPANISH9
db._base.collationfile=spa1252.df
```

The help for these settings:

Property	Description
db. codepage	The codepage of the database. Default: utf-8
db. template	The name of a sub-directory under '\${openedge.dir}/prolang' that contains resources that are used when creating the database and loading the collation tables. Ex. utf, dut, sch Default: utf
db. collation	The collation of the database (e.g. "basic", "icu-uca"). Default: icu-uca
db. collation file	The collation file to load into the database when it is created. If a relative path is assigned it will be resolved relative to the 'template' directory. When the file is not defined, the system will attempt to find an appropriate collation file by looking for files in the following order: [TEMPLATE]/[COLLATION].df [TEMPLATE]/_tran.df If a matching collation file is not found a hard error is raised when the database is created.



The codepage and collation 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 installer.)

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/qa/build/catalog/packages/fin-ee2016/2016/0/16/209
/schema/qadfin/finempty.df
schema.fin-qaddb-qadcpl.version=2016.0.16.209
...
```



The compile database (db.qadcpl) is used when compiling operational source code. Schema files configured for the main database (e.g. schema.INSTANCE.database=db.qaddb) will automatically be mirrored in the compile database. The compile database has the same schema as the main database, excluding the schema associated with QAD Reference Architecture (QRA) modules.

To display the help for schema settings:

```
> yab help schema.
```

SETTINGS

schema.area.map defined	Used to map the AREA(s) defined in the schema to the AREA(s) in the target database. Mapping cannot be performed on incremental schema files.
schema.centuryoffset the 100-year period	Century Year Offset (-yy) is used to determine the start of in which a date with a two-digit year is valid. Default: 1950
schema.cpsstream	The codepage to use (cpsstream) when reading the schema files. Default: utf-8
schema.database applied.	A reference to the database where the schema should be applied.
schema.dateformat	The date format (d) to use. Default: mdy
schema.dir	The root directory containing the schema files to process. Use either 'file' or 'dir' to locate schema files.
schema.excludes files	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 files	A comma-delimited list of include patterns to match schema files in the configured directory. Default: **/*.df
schema.numdec	The decimal separator (numdec) to use. Default: 46 (.)
schema.numsep	The number separator (numsep) to use. Default: 44 (,)
schema.objects.excludefiles exclude, one per line.	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 include, one per line.	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.
schema.renamefile have been renamed. Without a rename	A Progress rename file to describe tables and fields that have been renamed. Without a rename file, renaming a field A to B would result in dropping A and then adding B which could result in the loss of data. Syntax: T,[OLD TABLE NAME],[NEW TABLE NAME] F,[TABLE NAME],[OLD FIELD NAME],[NEW FIELD NAME] Ex. Rename table "foo" to "foo2" and field "bar" (formerly in "foo") to "bar2". T,foo,foo2 F,foo2,bar,bar2

Using Demo Data



QAD does not recommend upgrading Enterprise Edition in an environment where demo data is configured because this can lead to subsequent data load errors or loading of invalid data.

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

Verify current configuration

Demo data is distributed in a package whose name starts with "demo-data". You can verify that the environment does not already have the demo data installed with the following query:

```
> yab info | grep demo
```

Load Demo Data

If you do not have demo data installed and want to load it follow these steps.

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 QAD Enterprise Edition installer is used to reconfigure the environment to use the demo data.

```
> yab install [ENTERPRISE EDITION 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 or Installing Release data

To restore the original data or install the Release/Empty data follow these steps.

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

If you have an initial backup restore at this point

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

Rebuild Databases & Update

If you do not have an initial backup rebuild the databases to load Release data.

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

Using Parameter Files

Overview

Historically QAD has defined database connections using parameter files. When a Progress session starts, the connections defined in a parameter file are made under the authority of the user that installed Progress and not the user that started the Progress session. The Progress install user must have write permission on the database files when making a shared memory connection, but the user starting the session does not. Beginning with QAD Enterprise Edition 2015 (Cloud) and QAD Enterprise Edition 2016 (On-Premise), databases are connected from within a Progress session by the QAD Reference Architecture (QRA) bootstrapping procedure and not from a parameter file. The consequence of this change is that the user that starts a Progress session requires write permission to the databases connected with a shared memory connection.

This page describes how to configure Enterprise Edition to define database connections through parameter files.

Preparation

When parameter file connections are enabled, the system will create parameter files to connect the databases and remove database connection information from all QRA bootstrap files. By default, the parameter files have the same name as the bootstrap files but with a ".pf" file extension.

```
> yab config "bootstrap.*.file" "bootstrap.*.parameterfile"
bootstrap.default.file=/dr01/qadapps/qea/config/bootstrap.xml
bootstrap.default.parameterfile=/dr01/qadapps/qea/config/bootstrap.pf
bootstrap.foundation.file=/dr01/qadapps/qea/config/foundationbootstrap.xml
bootstrap.foundation.parameterfile=/dr01/qadapps/qea/config/foundationbootstrap.pf
```

Also when parameter file connections are enabled, the factory default configuration of standard Progress sessions will reference the parameter files connecting the databases. Converting a system to use parameter files to connect to the database can be transparent as long as these configuration settings are not overridden and/or the system does not have any third party or custom Progress sessions that need to be adjusted.

Settings that end with *svrstartupparam* configure server sessions to use parameter files and settings that end with *properties.pf* configure client sessions to use parameter files. The following commands will locate server and client settings that have been overridden.

```
> find build/config -name "*.properties" | grep -v "build/config/packages" | xargs grep
svrstartupparam
> find build/config -name "*.properties" | grep -v "build/config/packages" | xargs grep
properties.pf
```

Make a note of any overridden settings and the files in which they are defined.



If Enterprise Financials 2016, 2016.1, 2017, or 2017.1 is installed, a Financials patch may be required to support parameter file connections. Please review your Financials patch version before proceeding with the change.

Enabling

The *bootstrap._useparameterfile* setting is used to enable (or disable) the use of parameter files to connect databases.

build/config/configuration.properties

bootstrap._useparameterfile=true

If you identified any overridden settings in the previous step, they should be adjusted to reference the correct parameter file. The (-trace) option on the *config* command may be used show the value of the factory default setting that is being overridden. In the following example, the second definition of the setting *appserver.mfg.svrstartupparam* is being overridden by the first.

```
yab -trace config appserver.mfg.srvrstartupperam
appserver.mfg.srvrstartupperam=-mmax 20408 -Bt 20000 -tmpbsize 4 -nodupttidxerror

[1: build/config/system/fin2018-core-systest-pc.properties]
appserver.mfg.srvrstartupperam=-mmax 20408 -Bt 20000 -tmpbsize 4 -nodupttidxerror

[2: build/config/packages/yab-ee-foundation/1/10/0/10/default/yab-ee-foundation.properties]
# @if bootstrap._useparameterfile=true
# @if openedge.version =* [11.7.4, 12)
appserver.mfg.srvrstartupperam=-pf ${bootstrap.default.parameterfile} -mmax 16384 -Bt 10000 -
tmpbsize 4 -nodupttidxerror
```

The arguments "-pf \${bootstrap.default.parameterfile}" defined by the overridden setting need to be integrated into the first setting as in the following example:

```
appserver.mfg.srvrstartupperam=-pf ${bootstrap.default.parameterfile} -mmax 20408 -Bt 20000 -
tmpbsize 4 -nodupttidxerror
```

When the configuration of all Progress sessions have been adjusted to reference the correct parameter file, the changes are applied with the following commands which will stop the environment, reconfigure the environment, and then start the environment:

```
> yab stop
> yab reconfigure
> yab start
```

The *reconfigure* command will generate the parameter files, update the bootstrap configuration files, rewrite scripts, and adjust the Progress ubroker.properties configuration file.

Reconfiguring Tomcat

Enabling the Manager Web Application

Tomcat provides the manager web application to remotely control a Tomcat instance.

When Tomcat is installed in secure mode the manager web application is not deployed into the Tomcat instance.

Ex. Tomcat instance configured to use secure mode.

```
> yab config tomcat.default.enablesecure
tomcat.default.enablesecure=true
```

When Tomcat is not installed in secure mode the manager web application is deployed, but no accounts are preconfigured to allow access to the manager web application for security reasons. The "manager-gui" role is a Tomcat role with permission to access the manager web application.

Ex. Provide access to the manager web application with the username (admin) and the password (mfgpro).

build/config/configuration.properties

```
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
```

Ex. Enable the manager web application on the default Tomcat instance.

build/config/configuration.properties

```
tomcat.default.roles.manager-gui.rolename=manager-gui
tomcat.default.roles.manager-gui.members=admin
tomcat.default.users.admin.username=admin
tomcat.default.users.admin.password=mfgpro
```

To add additional users...

```
tomcat.[INSTANCE].roles.manager-gui.members=admin,[USER]
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]
```

To apply the change:

```
yab tomcat-[INSTANCE]-update tomcat-[INSTANCE]-stop tomcat-[INSTANCE]-start
```

Ex. Apply the change to the default Tomcat instance.

```
> yab tomcat-default-update tomcat-default-stop tomcat-default-start
```

Verify

Navigate to the following URL in a web browser and login as the user "admin" with the password "mfgpro".

`http://[HOST]:[PORT]/manager`

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"
```

To apply the change:

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

The change will be applied to the *setenv.sh* script in the Tomcat instance *bin* directory.

Ex. Default Tomcat instance

```
> cat servers/tomcat/bin/setenv.sh
```

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* that 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/qa/build/work/keystore/default.bin
```

Ex. Show the configuration of the default keystore.

```
> yab config keystore.default.*
keystore.default.file=/dr01/qadapps/qa/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 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 into the Keystore

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 to enable SSL

Configure the following setting to true to enable SSL on a tomcat instance:

```
tomcat.[INSTANCE].usessl=true
```

If you imported multiple certificates into the keystore you can use the following setting to identify the alias of the certificate to use:

```
tomcat.[INSTANCE].keyalias=[ALIAS]
```

To apply the change update the Tomcat instance.

```
yab tomcat-[INSTANCE]-update
```



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

Ex. Checking the SSL port of the default Tomcat instance.

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

Reconfiguring PROPATHs

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 Edition defines a set of base PROPATH definitions which are then used and extended 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 <i>propath.base</i> to provide a base PROPATH for operational sessions.
propath.fin	Extends <i>propath.base</i> to provide a base PROPATH for Finance sessions.

Analysis

The following command lists all of the PROPATH settings and may be used to locate the setting that controls the Progress session you are interested in adjusting.

```
> 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 config appserver.fin.propath
appserver.fin.propath

[1: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @ref propath.fin
appserver.fin.propath=

[2: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @ref propath.base
propath.fin=

[3: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @append
# @group 3
propath.fin=${dist.dir}/fin/patch,${appdir}/dist/fin

[4: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @append
propath.fin=${packages.fin-bldata.dir},${packages.fin-bin64-qadfin.dir}/qadfin.pl

[5: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @unique
propath.base=

[6: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @append
# @group 0
# @id base
propath.base=${appdir}/config

[7: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @append
# @group 3
propath.base=${netui.pro.dir}/com/mfgpro

[8: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @append
propath.base=.,${packages.qra-oe11.dir}/lib/qra.pl

[9: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @append
# @id mfg-rcode
propath.base=${code.mfg.dir},${code.mfg.dir}/us,${code.mfg.dir}/us/bbi

[10: build/config/packages/yab-ee-foundation/1/4/0/93/default/yab-ee-foundation.properties]
# @id mfg-customizations-rcode
# @append
# @group 2 -ict-customizations
propath.base=${code.mfg-customizations.dir}

[11: build/config/packages/yab-qra/1/4/0/52/default/yab-qra-webui.properties]
# @append
# @path
propath.base=${code.activity-feed.dir}

[12: build/config/packages/yab-qra/1/4/0/52/default/yab-qra.properties]
# @path
# @append
# @group 1
propath.base=${packages.qracore.dir}/qad.qra.core/bin/qracore.pl

[13: build/config/packages/yab-qxtend/1/4/0/25/default/yab-qxtend.properties]
# @append
propath.base=${code.qxi.dir}

```

Add a path

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

Standard Priority

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

Elevated Priority

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

The PATH ID is used to associate the path (or paths) with a name that can be used when ordering the PROPATH (see below), but it 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
4	Extensions
(undefined)	Standard

See the documentation for the *group* annotation in [Configuration Files](#) for more details.

Ex. Add some test programs to the operational compile.

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

Order Paths

The group assignment is used to define a first level ordering of the paths, where all group 0 paths will precede all group 1 paths which will precede all group 2 paths and so on. A secondary ordering of the paths within a group is afforded by ordering hints associated with the group annotation (see the documentation for the *group* annotation in [Configuration Files](#)).

Ex. Add some test programs to the operational compile with higher precedence than the "main" operational code.

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

The group order is used to take explicit control over the ordering of paths within a group. It will replace the ordering hints when defined. The paths must define an ID to be ordered by this mechanism.

Ex. Order the test paths ahead of the "main" operational paths.

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

See the documentation for the *grouporder* annotation in [Configuration Files](#) for more details.

Reconfiguring AdminServer

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

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

Reconfiguring Application Servers

Progress application servers are configured through the *appserver* type.

```
> yab config-help appserver
```

```
> yab -instances config appserver
```

Enabling and Disabling 4GLTrace

Enable

To increase the logging output from an application server configure the appropriate logging settings:

Ex. Configure extra logging on the 'mfg' application server

```
appserver.mfg.srvrlogginglevel=4
appserver.mfg.srvrlogentrytypes=ASPlumbing,DB.Connects,4GLTrace
```

And then apply the change to 'build/work/generated/ubroker.properties':

Ex. Apply the configuration change to the 'mfg' application server

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



If the environment is online the change will only have an effect if allowruntimeupdates is enabled for the application server.

```
> yab config appserver.mfg.allowruntimeupdates
appserver.mfg.allowruntimeupdates=1
```

If the brokers are very busy, it may take a while for a change to 4GLTrace (enabling and disabling) to have an effect.

Disable

With 4GL Tracing enabled, the application server will write a lot more information to its log files and this can slow down the application and consume excessive amounts of disk space, so it is generally a good idea to disable 4GLTrace after capturing the data to evaluate. To undo the change, remove the settings from *configuration.properties* (or comment them out) and then apply the changes.

Ex. Revert back to the original logging configuration on the 'mfg' application server

```
# appserver.mfg.srvrlogginglevel=4
# appserver.mfg.srvrlogentrytypes=ASPlumbing,DB.Connects,4GLTrace
```

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

Migrating to a New Host

Preconditions

To migrate an environment to a new host using this procedure the following preconditions must be met:

- The paths locating environment files should be the same on the source and destination host.
- The operating system should be the same on the source and destination host.

Procedure

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

```
> yab stop
```

Step 2: Copy all environment files to the destination host (if files are on a local file system).

Step 3: Configure the host setting on the destination host.

```
build/config/configuration.properties
```

```
host=[DESTINATION HOST]
```

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

```
> yab update restart
```

TCP/IP Ports

Servers use TCP/IP ports for communication.

By default ports are allocated to servers from ranges configured by the *ports* and the *dynamic-ports* settings.

```
> yab config ports
ports=22000-22100

vmwtb02:sc2> yab config dynamic-ports
dynamic-ports=1026-2000
```

Servers that require a stable port are allocated ports from the *ports* range. Once a port is allocated to a server it will continue to be assigned to that server. The file *build/work/system/ports* records port allocations. Servers that dynamically allocate ports are configured with the *dynamic-ports* range.

See the documentation for the *port* annotation in [Configuration Files](#) for further details.

Explicit Port Assignment

Alternatively ports can be directly assigned to servers.

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

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

The statement "*# @port*" in the example above is a port annotation. The annotation simply serves to let the system know that the port number (22001) has been assigned and should not be allocated to any other server without an explicit port assignment.

To explicitly configure dynamic port ranges use the information in the following table:

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

It may also be necessary to configure a new instance of the Language configuration type to update application files like */o cale.dat*, *encoding.dat*, and *configscreens.xml*.

```
> yab help "language."
```

SETTINGS

language.encoding	The encoding info for the language with the format: [MFG/PRO language code], [Java encoding], [Progress encoding], [html/xml encoding], [xsl encoding] e.g. us, ISO8859_1, ISO8859-1, utf-8, utf-8
language.id	The id for the language
language.locale	The locale info for the language with the format: [MFG/PRO language code], [ISO language code], [ISO Country code], [optional variant], [date format], [numeric format], [Progress language code] e.g. US, en, US,, mdy, American, ame
language.program.INSTANCE.defaulttable	The default table of the program
language.program.INSTANCE.name	The language might need to update the program section in configscreens.xml file, then define the new program here will do that trick The name of the program
language.program.INSTANCE.tables	The table list of the program

If the new language should add additional codepages to the Financials *server.xml* file then append a value to the setting `additional.codepages`.

Ex. Defining Thai Language

build/config/configuration.properties

```
language.th.id=th
language.th.locale=US,en,TH,,dmy,Thailand,tha
language.th.encoding=${language.th.id},windows-874,620-253,windows-874,windows-874

language.th.program.kasoivmt.name=kasoivmt.p
language.th.program.kasoivmt.defaulttable=so_mstr
language.th.program.kasoivmt.tables=so_mstr,sod_det

# @append
additional.codepages=620-253
```

Apply Changes

Run an update to add the language.

```
> yab update
```

Using AIA

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

```
build/config/configuration.properties
```

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

```
> yab tomcat-default-stop reconfigure tomcat-default-start
```



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

By default, connections from AppServer clients will be restricted to those that connect using HTTPS tunneling, when the Tomcat that hosts AIA is configured to support SSL (i.e. `tomcat.default.usessl=true`). This restriction can be explicitly enabled or disabled using the `aia.default.httpsenabled` setting.

Ex. Explicitly enable HTTPS

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

SSH and Telnet Protocols

SSH

By default the .NET connection manager and terminal mode programs are configured to use the SSH protocol, because it is more secure than the Telnet protocol.

To run programs in the .NET client in terminal mode, however, the client must have the Granados SSH library for .NET (Routrek.Granados.dll) installed. If the *Routrek.Granados.dll* file is copied into the Home Server web application directory, clients will download the DLL as necessary when they connect to the Home Server. Use the following command to determine the location of the Home Server web application.

```
> yab config webapp.homeserver.dir
```



QAD cannot redistribute the Granados SSH library due to export restrictions. This library can be downloaded from:

<http://granados.sourceforge.net/>

Telnet

Use the following configuration to switch from the SSH protocol to the Telnet protocol:

build/config/configuration.properties

```
netui.connmgr.protocol=telnet
netui.connmgr.port=23
netui.telnet.protocol=telnet
netui.telnet.port=23
```

To apply the change:

```
yab reconfigure restart
```

Shell Configuration

The following settings are used by the .NET connection manager and terminal mode programs when connecting to the server as the OS account configured by the *netui.connmgr.user* and the *netui.telnet.user* respectively (typically the same account). These settings should match the prompts that are produced when making an SSH or Telnet connection to the server interactively as the OS account.

Setting	Description
netui.connmgr.serverprompt	The shell prompt, which is dependent upon the default shell, and can vary based on user configuration, as well as across different operating systems. Default: \$
netui.connmgr.loginprompt	The prompt for the user name. Default: login:
netui.connmgr.passwordprompt	The prompt for the password. Default: Password:

To apply:

```
> yab reconfigure
```

Bootstrap Files

Application sessions are configured with bootstrap files, which in turn are configured by the bootstrap configuration type.

```
> yab config "bootstrap.*"
bootstrap.default.database.network=false
bootstrap.default.databases=db.qaddb,db.qadadm,db.qadhlp,db.qadmodule,db.qxevents
bootstrap.default.excludes=eambridge
bootstrap.default.file=/qad/sandbox/user/wtb/02/sc2-core/config/bootstrap.xml
bootstrap.default.global-module-locator=/qad/sandbox/user/wtb/02/sc2-core/config/global-module-locator.xml
bootstrap.default.modules.base.databases=db.qaddb
bootstrap.default.modules.mfgcoreplus.databases=db.qaddb
bootstrap.default.programs.0.name=mfaistrtp
bootstrap.foundation.database.network=false
bootstrap.foundation.databases=db.qaddb,db.qadadm,db.qadhlp,db.qadmodule,db.qaddb,db.qadadm,db.qadhlp,db.qadmodule
bootstrap.foundation.file=/qad/sandbox/user/wtb/02/sc2-core/config/foundationbootstrap.xml
bootstrap.foundation.global-module-locator=/qad/sandbox/user/wtb/02/sc2-core/config/global-module-locator.xml,/qad/sandbox/user/wtb/02/sc2-core/config/global-module-locator.xml
bootstrap.foundation.includes=qracore,mfgcoreplus,fincore
bootstrap.foundation.modules.mfgcoreplus.databases=db.qaddb,db.qaddb
```

A new bootstrap file can be defined based on the configuration of one of the standard bootstrap files. For example, to create an alternate version of the default bootstrap file that connects to the configured databases over the network, possibly to support clients in self service mode, you could configure the following:

build/config/configuration.properties

```
# @extends bootstrap.default
bootstrap.default-network=
bootstrap.default-network.database.network=true
bootstrap.default-network.file=${qra.config.dir}/bootstrap-network.xml
```

And this configuration would define a new process to generate the bootstrap file during an update:

```
> yab bootstrap-default-network-update
> cat config/bootstrap-network.xml
...
```

To use the alternate bootstrap in a client script, the script would pass the location of the bootstrap file (on the PROPATH) to the ClientBootstrap.p program as demonstrated below:

```
$DLC/bin/_progres -pf /qad/sandbox/user/wtb/02/sc2-core/build/work/generated/application.pf
$STARTUP_PARAMS -cpinternal $CODEPAGE -cpstream $CODEPAGE -p com/qad/qra/core/ClientBootstrap.p -
param bootstrap=bootstrap-network.xml,startup=mf.p,logfile=/qad/sandbox/user/wtb/02/sc2-core/build
/logs/client/session-chui-${USER}.log
```

Database Connections

Historically database connection information was defined in bootstrap files, but new installations are setup to define database connection information in parameter files instead. The setting `bootstrap._useparameterfile` controls whether connection information is stored in bootstrap files or in parameter files.

```
yab help bootstrap._useparameterfile
```

SETTINGS

bootstrap._useparameterfile parameter file	When true, database connections will be defined in a instead of the bootstrap file.
processed requires defined	When databases are defined in a parameter file they are as startup parameters and a shared memory connection only read permission on the database files. When databases are in a bootstrap file they are processed at runtime using
CONNECT write bootstrap file.	statements and a shared memory connection requires read and permission on the database files. There is no advantage to defining connections in the The default is false for backwards compatibility. This is NOT an instanced setting.
files.	Ex. Use PF files to connect databases for all bootstrap bootstrap._useparameterfile=true
without Progress bootstrap's parameter use file.	WARNING: The value of this setting should not be changed understanding the implications. When the value is true, all sessions that use bootstrapping MUST reference the file and when the value if false, all Progress sessions that bootstrapping MUST NOT reference the bootstrap's parameter
respect this sessions	The factory default settings for standard EE sessions option, but overrides (i.e. srvrstartupparam) and custom should be reviewed for compatibility. See: bootstrap.INSTANCE.parameterfile Default: false

Environments that are upgraded are not automatically updated to use parameter files because switching to use parameter files could break the environment and requires a careful review of all settings that are not set to their factory default value.

See: [Using Parameter Files](#)

QAD Reporting Framework Printers

QAD Reporting Framework printers are configured using instances of the `qrfrprinters` configuration type.

```
> yab help qrfrprinters

SETTINGS

    qrfrprinters.description      The QRF Printer's Description. Ex. qrfrprinters.1.
description=Kevin printer

    qrfrprinters.uncpath          The QRF Printer's UNCPath. Ex. qrfrprinters.1.
uncpath=\\corp23\\rm01
```

Ex. Define two printers.

```
build/config/configuration.properties

qrfrprinters.1.uncpath=\\corp23\\rm01
qrfrprinters.1.description=Mark's Favorite Printer

qrfrprinters.2.uncpath=\\corp23\\rm02
qrfrprinters.2.description=Andy's Favorite Printer
```

To apply:

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

Batch Jobs

A batch job executes a program in a non-interactive application character session where input is read from a file. Batch jobs are used to automate character session tasks. Two different approaches to managing batch jobs are supported. The [Batch Jobs \(Enhanced\)](#) approach has more features and was introduced as an enhancement to the original [Batch Jobs \(Basic\)](#) approach. The [Batch Jobs \(Basic\)](#) approach has been retained for backwards compatibility and is more than sufficient for certain use cases.

Input

In both cases, input is fed to the Progress session through an input file which uses the following characters to represent control keys:

#	Character	Control Key	Notes
#	-----	-----	-----
#	-	TAB	(Single line per frame)
#	{ENTER}	F1	
#	.	F4	(On a new line)

Ex. Login, run Item Master Maintenance, and exit (multiple domains choosing default domain for user)

```
mfg -
1.4.1
.
.
Y
```

Ex. Login, run Item Master Maintenance, and exit (single domain)

```
mfg -
1.4.1
.
.
Y
```

Batch Jobs (Basic)

This approach uses the script `batch.[sh|py]` to run the batch program. The script takes three arguments:

Arg	Description	Example
1	The code page for the session.	utf-8
2	The batch program to run.	program.p
3	Arguments to add to the Progress (-param) argument. NOTE: The application will also embed comma-delimited data in the (-param) argument, for example, using <code>startup=[PROGRAM]</code> to locate your batch program.	foo=bar, bar=baz

Here is an example batch program that prints out the (-param) data and then runs the application.

customizations/mfg/default/src/program.p

```
define variable i as integer no-undo.
define variable element as character no-undo.
do i = 1 TO num-entries(session:parameter):
    put unformatted entry(i, session:parameter) skip.
end.

run mf.p.
```

To compile the batch program:

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

To run the batch program with [input](#) from a file named input.

```
scripts/batch.[sh|py] utf-8 program.p < input | tee
```

The `batch.[sh|py]` script is configured by the following settings:

```
> yab config filegen.foundation-batchclient.*
```

Batch Jobs (Enhanced)

This approach configures batch jobs with instances of the batch configuration type. For every batch instance defined, the system will define a command that executes a system generated script to process the batch job.

```
# A comma-delimited list of files providing input for the batch session.
# Relative paths will be resolved relative to the working directory.
# Files can use the following characters to represent control keys.
#
#      Character      Control Key      Notes
#      -
#      -              TAB              (Single line per frame)
#      {ENTER}        F1              (On a new line)
#      .              F4
#
batch.in=

# The file where output should be written.
# A relative path will be resolved relative to the working directory.
# Default: batch.out
batch.out=

# To add a timestamp suffix to the output file.
# Default: false
batch.out.timestamp=

# The working directory to use for the batch session.
batch.workdir=

# Progress startup parameters to add to the startup parameters for the
# batch session.
batch.progress.params=

# A comma-delimited list of application parameters (KEY=VALUE, KEY=VALUE...)
# to add to the (-param) startup argument for the batch session.
batch.application.param=

# A lock file to control batch session concurrency.
# The lock file will be created before starting the batch session and removed
# after the batch session completes. If the lock file already exists the
# batch session will not be started.
batch.lockfile=

# An optional program to run before the system runs mf.p to start a character session.
batch.initprogram=
```



The system will run mf.p to start a character session with the input you have configured. The optional *initprogram* setting **sh** **could not** be configured with the value *mf.p* and if another program is configured it should not run *mf.p*.

Here is an example init program that prints out the (-param) data.

customizations/mfg/default/src/program.p

```
define variable i as integer no-undo.
define variable element as character no-undo.
do i = 1 TO num-entries(session:parameter):
    put unformatted entry(i, session:parameter) skip.
end.
```

Ex. Configure a new batch job (job1).

```
batch.job1.initprogram=program.p
batch.job1.workdir=${appdir}/batch/job1
batch.job1.in=../login.in,job1.in,../logout.in
batch.job1.out=job1.out
batch.job1.out.timestamp=true
batch.job1.application.param=foo=bar,bar=baz
```

Ex. An example login sequence, logging in as the default "mfg" user.

login.in
mfg -

Ex. An example logout sequence (there is an empty line at the end).

logout.in
. . Y . Y

Ex. Process the configuration.

```
> yab batch-script

> ll scripts/batch
total 8
-rwxrw-r-- 1 wtb devel 498 Mar 28 10:06 batch-job1

> yab help batch-job1-execute
PROCESS
    batch-job1-execute - Executes the [job1] batch job.

> cat scripts/batch/batch-job1
#!/usr/bin/python
# Runs the batch.job1 batch job.
# Generated: 2017-10-04T05:59-0700
import os
os.environ['APPLICATION_PARAM_APPEND'] = r"batchIn=../login.in,batchIn=job1.in,batchIn=../logout.
in,batchOut=job1.out,batchOutTimestamp=true,initProgram=program.p,foo=bar,bar=baz"
os.environ['WORKDIR'] = r"/qad/sbox/004/user/wtb/02/sr2-core-systest/batch/job1"
os.environ['STARTUP'] = r"batch.p"
os.environ['PROGRESS_PARAMS_APPEND'] = r"-b"

import subprocess
import sys
params = [r"/usr/bin/python", r"/qad/sbox/004/user/wtb/02/sr2-core-systest/scripts/client-xx"]
p = subprocess.Popen(params , stdout=subprocess.PIPE, stderr=subprocess.PIPE)
out, err = p.communicate()
print(out)
if err is not None:
    print(err)
if p.returncode != 0:
    sys.exit(p.returncode)
```

Configuring Commands

The *process* configuration type 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]
```

In the following example we define a new command database-backup-setup to execute a script and another command database-backup-cleanup to execute another script. We then "anchor" the setup command to run before the existing database-backup command and the cleanup command to run after it.

build/config/configuration.properties

```
process.database-backup-setup.id=database-backup-setup
process.database-backup-setup.args=/dr01/scripts/backup-setup
process.database-backup-setup.impl=exec
process.database-backup-setup.anchorbefore=database-backup

process.database-backup-cleanup.id=database-backup-cleanup
process.database-backup-cleanup.args=/dr01/scripts/backup-cleanup
process.database-backup-cleanup.impl=exec
process.database-backup-cleanup.anchorafter=database-backup
```

And then we can check our work by listing the commands that would be executed when the database-backup command is executed:

```
> yab system-process-list database-backup

1. database-backup-setup
2. database-backup
3. database-backup-cleanup
```

See [Exec Commands](#) for more elaborate examples of configuring commands to call programs.

In the following example we define a new command compile as a convenience to recompile the standard and customized operational code and then trim the MFG application server.

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

Ex. Display help for the *process* configuration type.

```
yab help "process."
```

Exec Commands

A command to invoke an executable program or script is defined by setting the *imp/*setting of the process to "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 (preferably)

```
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.greeting.anchormeafter=start
```

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)
```

ANT commands

A command to run an [Apache ANT](#) script is defined by setting the *impl* setting of the process to "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.ant-test.anchorbefore=update
```

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
```

Error Handling

When a request is submitted ([update](#) for example), it is evaluated into an ordered sequence of commands. If one of the commands reports an error the request will be aborted. There are two ways to alter the default request processing behavior, you can tell the system to continue processing the request if a process reports an error with the *failonerror* setting or you can tell the system to always execute certain commands even when the request is aborted using the *always* setting.

Failonerror

The *failonerror* setting is used to configure whether or not errors should fail (abort) the request. The default setting is true which means that any error will abort the request. The most typical use for this setting is as a request option to address a transient problem. For example, in an environment where a server cannot be stopped, *failonerror* might be added to the *stop* request to continue to shut down the remaining servers in the normal manner.

Ex. Continue to process all stop commands even if one or more report errors.

```
> yab -failonerror:false stop
```

It is also possible to only allow failures from specific commands using the syntax:

```
-failonerror.[PROCESS]:false
```

Ex. Do not halt the update if 'qxtend-default-update' or 'event-service-update' report errors.

```
> yab -failonerror.qxtend-default-update:false -failonerror.event-service-update:false update
```

The *failonerror* handling can be more permanently associated with a process through configuration. For example, a process that was responsible for sending an inessential notification to an unreliable service might be configured so it will never abort the request.

Ex. Try and send the notification but don't abort the request if it fails.

```
process.notify.id=fire-and-forget-notification
process.notify.failonerror=false
...
```

All errors reported by commands are recorded in the YAB log file regardless of how *failonerror* is configured.

Always

The *always* setting is used to configure commands that should be processed even when the request is aborted. When the *always* setting is enabled for a command it applies to that command as well as the commands associated with it.

The *always* setting may be used as a request option where it is scoped to the request. Consider the following scenario where a hypothetical custom-backup command will turn off environment monitoring, stop the environment, backup all databases, start the environment, and turn monitoring back on. If any of the database backups fail we may not want to attempt further backups, but we would like to bring the environment back online and re-enable monitoring. Furthermore we only want to always execute the *start* command in the context of this backup process. In the context of an *update* we do not want to start the environment if the *update* fails.

```
> yab -always:start -always:monitoring-start custom-backup
```

The *always* handling can be more permanently associated with a process through configuration. Extending the previous example, we might say that it is appropriate for the *monitoring-start* process to always run when it is used in any context.

```
# A hypothetical process whose failure should not abort the request.
process.monitoring-start.id=monitoring-start
process.monitoring-start.always=true
...
```

All errors reported by commands are recorded in the YAB log file regardless of how *always* is configured. The last error is used to abort the request when all *always* commands have been processed.

Timeouts

Timeouts are used to control the maximum amount of time the system should wait for some outcome. Timeouts are configured with an inheritance algorithm for finding a fallback timeout value if the more specific timeout configuration is undefined. For example, the timeout setting "timeout.foo.bar" would be resolved to "timeout.foo.bar" and if not set to "timeout.foo" and if not set to "timeout".

All "timeout.*" settings are expressed in milliseconds.

When a timeout is exceeded an error is raised.

```
com.qad.build.utilities.CommandExecutor$TimeoutException:  
The task has exceeded the configured timeout of [0.001 s].  
This timeout may be individually configured by the [timeout.adminserver.stop] property and  
currently is defaulted by the [timeout] property.
```

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 the configuration of packages (i.e. the settings that begin with "packages") should be reevaluated.

With the default configuration, the system will automatically reevaluate the configuration of packages as needed. Setting (check-for-updates=true) forces the configuration of packages to be reevaluated whenever the configuration (package or otherwise) changes. Setting (check-for-updates=false) prevents the reevaluation of the configuration of packages.

A production environment may be configured with (check-for-updates=false) to ensure that software in the environment is only updated with a specific acknowledgement. When the system is configured with (check-for-updates=false), the setting can be temporarily overridden to process package updates as follows:

```
> yab -check-for-updates [COMMAND]
```

Ex.

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

The `install` command will automatically force the reevaluation of packages. When `(-check-for-updates)` is configured as a request option, it implies a [refresh](#) `(-r)`.

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 system 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 in the file:

build/config/system/loader.properties

dependency settings

A family of settings to fine tune the evaluation of [package settings](#).



To list the help for all package evaluation settings:

```
> yab help dependency.
```

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 colon ':' 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 colon ':' 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 colon ':' 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 colon ':' delimited list of dependency redirections.

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

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

```
dependency.redirects=abc=abc-3.4.12
```

dependency.types

Configures the package constraint types that will be evaluated.

The value is a "bitwise" combination of the following types:

Type	Description
1	Grouping constraints
2	Installation constraints
4	Runtime constraints

Default: 7 (all)

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)
```

Conditions

Packages can be configured so they are included only when certain conditions are met.

There are two types of conditions which are distinguished in part by when they are evaluated, either before or after the package configuration is resolved.

Before Resolution

The before resolution conditions are evaluated before packages are resolved. A package with an unsatisfied condition will not be included in the package resolution request.

Include the package if any of the package classes are configured.

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

Include the package if all of the package classes are not configured.

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

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,collaboration
```

After Resolution

The after resolution conditions are evaluated after packages are resolved. A package with an unsatisfied condition will be staged but disabled.

Enable or disable the package.

```
packages.NAME.enabled=[true|false]
```

If 'enabled' is not set 'enabled.if' and 'enabled.unless' conditions will be evaluated (if defined).

Enable the package if any of the package ranges are configured.

```
packages.NAME.enabled.if=PACKAGE RANGE; PACKAGE RANGE...
```

Enable the package if all of the package ranges are not configured.

```
packages.NAME.enabled.unless=PACKAGE RANGE; PACKAGE RANGE...
```

Ex. Enable the 'mongodb' package if either a 3.x version of the 'event-service' or the 'collaboration' package is configured.

```
packages.mongodb=mongodb-rhel5-3.0.3.0
packages.mongodb.enabled.if=event-service[3,4];collaboration[3,4]
```



The after resolution conditions use a semi-colon as a delimiter (because a comma is a valid character in a range expression), whereas the before resolution conditions use a comma.

Override

The 'override' setting is used to force all constraints for a package to use the configured value. It is the equivalent of defining a [redirect](#) statement for the package.

```
packages.NAME.override=[true|false]
```

Ex. Configure the 'base-1-4-0-139-customer-1.4.0.20004' hotfix package to override the standard 'base' package.

```
packages.base=base-1-4-0-139-customer-1.4.0.20004
packages.base.override=true
```

The override is equivalent to the following setting:

```
dependency.redirects=base=base-1-4-0-139-customer-1.4.0.20004
```

Patching

See [Package Patch](#).

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
```

The `dynamic-ports` setting is used to allocate a default port range for services that dynamically open ports.

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

See [TCP/IP Ports](#) for more details.

verify setting

Instructs YAB to verify commands.

This is best illustrated with an example. The `tomcat-default-start` command will start the default Tomcat instance if it is not already running. The command will execute a Tomcat script to start the server (see: `scripts/tomcat-default-start`) and this script is asynchronous, meaning it will begin to start the Tomcat instance but then immediately return control to the calling program. When the command returns a `STARTED` status this can be interpreted as saying that the Tomcat server was not running and a request was successfully submitted to start the server. The `(-verify)` option will take this one step further and wait for a configurable amount of time until the Tomcat server is started (Tomcat is listening on the HTTP or HTTPS port and all testable web applications are active), returning a `PASSED` status when the desired state is verified.

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

```
> yab -verify start
                                start (18 tasks)
-----
1/18  adminserver-start          STARTED (1.662 s)
      adminserver-start          PASSED  (2.980 s)
2/18  nameserver-start           OK (0.000 s)
      nameserver-start           PASSED  (51.648 s)
3/18  database-qadadm-start       STARTED (2.305 s)
      database-qadadm-start       PASSED  (0.018 s)
4/18  database-qaddb-start        STARTED (2.598 s)
      database-qaddb-start        PASSED  (0.026 s)
5/18  database-qadhlp-start       STARTED (2.262 s)
      database-qadhlp-start       PASSED  (0.019 s)
6/18  database-qxevents-start     STARTED (2.238 s)
      database-qxevents-start     PASSED  (0.016 s)
...

```

The `(-verify-no-exec)` option does the verification but without executing any commands. For example, the environment could be stopped without verification, and then subsequently tested using the `(-verify-no-exec)` option which might provide better overall performance.

```
> yab stop
...
> yab -verify-no-exec stop
                                stop (18 tasks)
-----
1/18  qxtend-stage                SKIPPED (0.000 s)
2/18  qxo-services-stop           SKIPPED (0.000 s)
3/18  tomcat-default-stop         SKIPPED (0.000 s)
      tomcat-default-stop         PASSED  (0.048 s)
4/18  tomcat-qxtend-stop          SKIPPED (0.000 s)
      tomcat-qxtend-stop          PASSED  (0.005 s)
5/18  webspeed-default-stop       SKIPPED (0.000 s)
      webspeed-default-stop       PASSED  (0.239 s)
6/18  appserver-fin-stop          SKIPPED (0.000 s)
      appserver-fin-stop          PASSED  (0.221 s)
...

```

The maximum amount of time to wait for the system to be in the desired state, the delay between status checks, and the handling of failures can be configured:

```
> yab help verify
```

SETTINGS

verify	True to run functional tests.
verify-no-exec	True to run functional tests while skipping the execution of all processes.
verify.maxwait	The maximum amount of time to wait for the condition to be satisfied as measured in seconds or 0 to wait indefinitely.
	This setting configures verifications (enabled with the -verify option) that wait for some condition to be true.
	NOTE: The 'maxwait', 'retryinterval', and 'timeoutexception' settings are configured using an algorithm where the first defined setting takes precedence. Using the 'database-qaddb-stop' process and the 'maxwait' setting as an example, the following settings would be evaluated in this order:
	verify.database-qaddb-stop.maxwait verify.database-stop.maxwait verify.maxwait
verify.retryinterval	The amount of time to wait after testing the condition before trying again.
	This setting configures verifications (enabled with the -verify option) that wait for some condition to be true. See 'maxwait' for more details on how verification settings are resolved.
verify.silent	When true the output from verification will be kept to a minimum.
	This setting configures verifications (enabled with the -verify option) that wait for some condition to be true.
verify.timeoutexception	True to throw an exception immediately if the condition is not satisfied before timing out.
	When false all verifications will be performed and the request will be failed if any fail.
	This setting configures verifications (enabled with the -verify option) that wait for some condition to be true. See 'maxwait' for more details on how verification settings are resolved.

Ex. Configure all verifications to wait at most for 120 seconds

```
build/config/configuration.properties
```

```
verify.maxwait=120
```

Ex. Configure all database stop verifications to wait at most for 120 seconds

```
build/config/configuration.properties
```

```
verify.database-stop.maxwait=120
```

Ex. Configure the QADDB database stop verification to wait at most for 120 seconds

```
build/config/configuration.properties
```

```
verify.database-qaddb-stop.maxwait=120
```



Only some commands have associated verifications.

Upgrades, Customizations, and Patches

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

Product Upgrades

Product updates are delivered as zip files. The zip file contains [packages](#) to install into the environment. Always review the product installation guide before upgrading the product.

To install the product:

```
> yab install product-[VERSION].zip
```

Delayed Install

The option (-install-update:false) is used to configure a "delayed install" of the product where the packages are integrated into the environment but the environment update is delayed.

```
> yab -install-update:false install product-[VERSION].zip
```

This delay can be used to "batch" install multiple products and then apply all of the products with a single update.

```
> 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 c* ommand.

```
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, which defaults to:

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 into the *data* directory.



As a convenience a *default* customization is pre-defined:

```
> tree customizations/mfg/default
customizations/mfg/default/
|-- data
|   |-- qadadm
|   |-- qaddb
|-- src
```

Compiling

To compile customized programs execute the command:

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

The r-code will be written into the the directory configured by the *code.mfg-customizations.dir* setting, which defaults to:

dist/mfg-customizations



To recompile a standard program using the customization compile (for example if the standard program references an include file that has been customized) add the program to the following file with a *propath* relative path (e.g. 'us/so/sosomt.p'):

```
customizations/mfg/compile.lst
```

If the customization includes data, execute the *update* command instead, to load the data and compile the programs:

```
> yab update
```



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-customizations.propath* setting to fine-tune the order. Customizations that are not enumerated in the *grouporder* annotation will be ordered after the enumerated customizations.

Ex.

```
# @grouporder 2 cust2 cust1
code.mfg-customizations.propath=
```

Removing

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

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

Integrate QAD source file archive into environment.

Operational source code that has been ordered is downloaded from the QAD Download Center.

The following steps describe how the archive downloaded from the QAD Download Center can be integrated into a YAB environment, and how source files can be copied to a customization area.

In the context of these source files, 'integrated' means the sources are in the environment catalog. It does not mean that the sources appear on any environment propath (compile or runtime).

A command is provided to move sources from the environment catalog into a customization area - which does appear in application propaths.

As a starting point it is assumed that the media has already been downloaded. Ex. File 00098765.125.7z has been downloaded to /dr01/mymedia.

Install source archive into environment catalog.

This step creates a YAB compatible package named mfg-customer-src and installs it in the environment catalog.

```
> yab customer-src-integrate -archive:/dr01/mymedia/00098765.125.7z
```

Upon completion a new package mfg-customer-src is added to the local catalog.

```
> yab info
...
|- mfg-customer-src                2018.0.18.191  local  (new)
...
```

Copy sources to customization area

Running 'customer-src-integrate' also sets up a copy command - customer-src-copy - to copy sources to the default customization area.

Note - when specifying wildcards (ex. *) as part of the copy the argument should be within quotes.

Ex. copy us/pt/* and us/ud/* to default customization area.

```
yab customer-src-copy "us/pt/*" "us/up/*"
```

Ex. copy us/wo/wosc*.i to customization area 'domain1'.

```
yab customer-src-copy -dest-dir:/path/to/env/customizations/mfg/domain1/src "us/wo/wosc*.i"
```

Verify code matches

Before making any changes to the files copied to the customization area verify source files match the files in the environment.

Use 'code-mfg-customizations-locate' to verify the code copied to the customizations area matches the code it is replacing.

Ex. verify file us/pt/ptsurp.p copied into customization area matches the original version of the file.

```

> yab code-mfg-customizations-locate -xcode us/pt/ptsurp.p
File
Modified          Hash (X)                      Enc
-----
-----
/dr01/qadapps/2019/customizations/mfg/default/src/us/pt/ptsurp.p      2018-11-21 05:13
AM 60165690907638bfddc7f918971168f9 No
/dr01/qadapps/2019/build/catalog/packages/mfg/2018/0/18/711/src/us/pt/ptsurp.p 2018-11-21 05:14
AM 60165690907638bfddc7f918971168f9 Yes

```

As the original file is encrypted we use the -xcode option to create an encrypted copy of the new file and.

If the hash of both files is not the same the version of the source archive must be reviewed to determine why they do not match.

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-customization/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/build/catalog/packages/qracore/2
/17/0/32/qad.qra.core/bin/qracore.pl,/dr01/qadapps/qea/dist/fin/patch,/dr01/qadapps/qea/dist/fin,
/dr01/qadapps/qea/dist/pro/com/mfgpro,/dr01/qadapps/qea/build/catalog/packages/fin-bldata/2016/0
/80/5,/dr01/qadapps/qea/build/catalog/packages/fin-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
```

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
```

Integrated Customization Toolkit

The Integrated Customization Toolkit (ICT) is used to develop customizations for QAD Enterprise Edition. If ICT is included in the environment, customizations developed with ICT are staged in the location configured by the *customization.s.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

Prerequisite



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

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

Installation

If the environment will be used to develop ICT customizations *ict.toolkit* should be set to true (default:false).

build/config/configuration.properties

```
ict.toolkit=true
```



Setting `ict.toolkit=true` configures application sessions with additional PROPATH entries to support the ICT development. These additional PROPATH entries should not be required when customizations developed with ICT are integrated into non-development environments.

To install ICT from the ICT installation media:

```
> yab install [ICT INSTALLATION MEDIA]
```

Or to install ICT from a software catalog:

build/config/configuration.properties

```
packages.integrated-customization-toolkit=[VERSION]
```

```
> yab update
```

Enter Toolkit License (Development Environments)

Start a character client session

```
> scripts/client-us
```

Register the toolkit license

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.

Grant ICT menu permissions in a .NET UI session

Run Role Permissions Maintain and update permissions to ICT menus.

Processes X Role Permissions X

Actions Setup Cancel [Navigation Icons] Add to Favorites Stored Searches

Role Name equals [Search] + -

Role Description equals [Search] + -

Active equals [Search] + -

Viewing 1 - 11 of 11 Records per page: 100

Role Name	Role Description
_Everyone	Role created by co
CustomerNotify	Create of customer
EmployeeNotify	Create of employee
EndUserNotify	Create of enduser
Grp1	Group1
PostAutoBalNotify	Create of Autobal p
qadadmin	Admin Group
rptAdmin	Report Administratc
rptDsgn	Report Developer
SuperUser	SuperUser Role
SupplierNotify	Create of supplier

Go To Actions Tools Print Preview Attach

Role Name SuperUser

Role Description SuperUser Role

Filter

- ☒ Secured items on menu
 - ☒ Customer Management
 - ☒ Supply Chain
 - ☒ Manufacturing
 - ☒ Financials
 - ☒ Master Data
 - ☒ System Administration
 - ☒ System Administration (36)
 - ☒ QAD ICT Development Kit (90)
 - ☒ Processes
- ☒ Secured items not on menu



The install creates the `ictdmp-ict-update` process to load ICT data using CIM. The process will not succeed (and will log a warning) when the core Enterprise Edition licenses have not been entered into the system. After the data is successfully loaded into the system, it cannot be reloaded (i.e. during a subsequent upgrade) until the application user configured by the user/password settings has been granted permissions to the ICT menus (see Grant ICT menu permissions above).

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 be 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 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
```

With the preceding configuration, the Progress native (.d) data files would be processed during the database - qaddb - data - dotd - update step in an update. 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.

Metadata

Metadata is another form of system data distributed in QAD Reference Architecture (QRA) modules. The `metadata-export` command is used to export metadata from a module to the file system.

Ex. Export Access Control Entries distributed by the `qracore` module.

```
> yab -type:ace metadata-export urn:module:com.qad.qracore
```

Process Maps

Customized process map content is staged in the *customizations/processmap* directory. Customized process map definitions can be added into the directory and additional sub-directories can be created as needed to organize the content. Property files added into the *properties* sub-directory will be merged into the deployed process map property files.

```
> tree customizations/processmap/
customizations/processmap/
|-- attachments
|-- images
`-- properties
```

As an alternative, the `webapp.pronav.content.excludes` setting may be used to exclude files from being updated by the system.

```
> yab help webapp.pronav.content.excludes

SETTINGS

    webapp.pronav.content.excludes    A comma-delimited list of patterns selecting content
files that should                    not be updated when new content is deployed into the web
application.                         Typically these patterns are used to protect customized
content from                         getting overridden with standard content.

customizations is to                An alternative (and better) approach to managing
                                     store them in the process map customization directory.

the web application root,           NOTE: The upgrade patterns should be defined relative to
                                     using forward slashes for nested paths (e.g. a/b/c).
```

The `webapp.pronav.upgrade.excludes` setting should be reviewed for accuracy. In conjunction with the `webapp.pronav.upgrade.includes` setting, it determines which files will be removed prior to an upgrade of the process map application. The factory default setting is typically correct for most environments:

```
> yab config webapp.pronav.upgrade.excludes
webapp.pronav.upgrade.excludes=attachments/**,images/**,WEB-INF/pronav/xml/**,WEB-INF/pronav
/properties/**
```

Adding Custom Database

A database 'extension.db' is created at install time. This is intended to be used for creating application extensions / customizations. By default the extension database is connected to most Progress runtime sessions. The 'Using' section describes how to connect this database to other sessions (ex. compile.)

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-${db.custom.physicalname}
dbserver.custom.databasesname=${db.custom.dir}/${db.custom.physicalname}
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.databasesname=/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/qa/customizations/database/custom.df
```

Using

Use the following configuration to include the database in all compile sessions:

```
# @append  
code._base.databases=db.custom
```

Or to add the database just to the operational compile:

```
# @append  
code.mfg.databases=db.custom
```

To include the database in most Progress runtime sessions:

```
# @append  
bootstrap.default.databases=db.custom
```

Update

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

```
> yab 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.
```

Compiling EDI Functions

EDI utilizes Progress functions (programs). A default compile for EDI sources is preconfigured in a new EE installation.

The default EDI compile is configured as follows:

- EDI sources are expected in dist/edifunc/default, and the rcode is also written to this directory.
- Compile database connections and propath are inherited from the code.mfg settings.
- The EDI compile is included as part of an update.

Using The Default EDI Compile

For EDI to use this default function location

- EDI functions must reside in dist/edifunc/default
- In eCommerceControl (35.13.24) the Function Directory field must be set to absolute path for dist/edifunc/default.
Ex. /qond/apps/mfgpro/dist/edifunc/default.

Compile EDI Functions Using YAB

EDI code is compiled automatically as part of an update. To compile all edi functions manually execute:

```
> yab code-edi-update
```

Add Additional EDI Function Locations

As the EDI functions may vary by domain, multiple EDI function locations may be required.

Ex. To add domain specific EDI function locations for Domain1 and Domain2.

1. Create directories dist/edifunc/domain1, dist/edifunc/domain2.
2. Create / copy required EDI functions in each domain specific EDI code folder.
3. Compile EDI functions as described above.
4. The rcode is written to each domain specific EDI code folder.

Adding Custom Directory Backup Processes

Custom directory backup commands can be configured. For example, the following configuration defines a new command to backup the WEB-INF folder for the desktop webapp, excluding the logs directory. This example writes the backups to a tag folder under `${dbbackup.dir}`.

build/config/configuration.properties

```
# @extends directorybackup._base
directorybackup.custom=${dbbackup.dir}
directorybackup.custom.source.dir=${webapp.desktop.dir}
directorybackup.custom.target.dir=${dbbackup.dir}
directorybackup.custom.includes=WEB-INF/**
directorybackup.custom.excludes=WEB-INF/logs,WEB-INF/logs/**
```

This configuration generates the following commands

directory-custom-backup	Backs up the 'custom' files.
directory-custom-backup-list files.	Lists the 'custom' backup
directory-custom-backup-remove files.	Removes the 'custom' backup
directory-custom-restore files.	Restores the 'custom' backup

Running Backup Commands

The backup command can either be run directly, or as part of other aggregating commands that call it. Ex. To run the 'custom' backup directly:

```
> yab directory-custom-backup
```

An aggregating backup command `directorybackup-backup` is available to group backups that should be run as part of an environment backup. By default a backup is not included in the aggregating command. To include a backup to run as part of the `directorybackup-backup` command:

```
directorybackup.custom.manual.backup=false
```

To list the backup commands registered with `directorybackup-backup`

```
> yab system-process-list directorybackup-backup
```

1. directory-action-center-backup
2. directory-custom-backup
3. directorybackup-backup

This aggregating command is run as part of commands `environment-offline-backup` and `environment-online-backup`.

Tags

The '-tag' option either gives a backup a tag to identify it, or specifies the backup to be restored or removed.

The `directorybackup.target.dir` setting defines the location where backups for this instance will be created. Within the backup directory, sub-directories based on the tag are used to organize backups. Previous backups that use the same tag are overwritten, unless configured to not overwrite

```
directorybackup.custom.override=false.
```

The default tag name is "default". Backups can be configured to use timestamps as the tag by default

```
directorybackup.custom.timestamp=true
```

The maximum number of timestamped backups that are stored kept can be limited, 0 implies no limit.

```
directorybackup.custom.timestampmax=5
```

Ex. Backup 'custom' to the "stable" tag.

```
> yab -tag:stable directory-custom-backup
```

Ex. Backup 'custom' to a timestamp tag.

```
> yab -directorybackup.custom.timestamp directory-custom-backup
```



When directorybackups are run as part of an environment-backup the the dbbackup.timestamp / dbbackup.timestampmax settings override the directorybackup.timestamp / directorybackup.timestampmax settings - ensuring a consistent timestamp across the environment-backup.

Listing backups

To list the available backups:

```
> yab directory-custom-backup-list
```

```
directory-custom-backup-list (1 task)
```

```
-----
1/1 directory-custom-backup-list
```

```
Tag: default
```

```
-----
Location: /qad/local/sandbox/backups/dvr/vmdvr02.qad.com/default/custom
```

```
Compressed: No
```

```
qadui/WEB-INF
```

```
Jul 18, 2018 9:57:25 AM
```

```
Tag: 20180718095548
```

```
-----
Location: /qad/local/sandbox/backups/dvr/vmdvr02.qad.com/20180718095548/custom
```

```
Compressed: Yes
```

```
qadui/custom.zip
```

```
Aug 31, 2018 7:19:56 AM
```

To list all backups for all directorybackup instances use `directorybackup-list`.

Restoring Backups

To restore a backup directly:

```
> yab directory-custom-restore
```

Similar to the aggregation of backup commands under an `directory-backups` command, restore commands are also aggregated under a `directorybackup-restore` command.



To prevent unintended restores each restore process must be configured to enable it run under the aggregating command using `directorybackup.manual.restore=false`

Ex 1. Run additional-restores without explicit enablement of any restore command.

```
>yab directorybackup-restore
```

```
directorybackup-restore (2 tasks)
```

```
-----
1/2 directory-action-center-restore      SKIPPED (0.001 s)
```

```
2/2 directory-custom-restore             SKIPPED (0.001 s)
-----
```

Ex 2. Run additional-restores enabling only `directory-action-center-restore` command

Configure `directorybackup.action-center.manual.restore=false`

```
> yab directorybackup-restore
```

```
directorybackup-restore (2 tasks)
```

```
-----
1/2 directory-action-center-restore      OK (1.048 s)
```

```
2/2 directory-custom-restore             SKIPPED (0.000 s)
-----
```

Use the '-tag' option to identify a specific backup to be restored. If '-tag' is not provided the tag 'default' is used.

Removing Backups

To remove a specific backup:

```
>yab directory-custom-backup-remove -tag:2018073119275657
```

If '-tag' is not provided the tag 'default' is used.

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
...
```

File Patch

A file can be patched using the *patch* configuration type. Typically a patch is used as a temporary corrective action to change a managed file in a way that is not yet supported by the system while ensuring that the change is not lost when the system is updated.

This is best illustrated with an example. Imagine it was necessary to add a *proserve* argument to the scripts that start database servers (i.e scripts/database-INSTANCE-start), but the argument was not yet supported by YAB. If the scripts were directly edited, it is possible the change would be lost the next time the system was [updated](#). A more durable solution would be to create a patch describing the change and then configure that patch to be applied after the database-script command executes (the command that is responsible for generating the standard script).

Creating the patch

Copy one of the start scripts

```
cp scripts/database-qaddb-start scripts/database-qaddb-start.modified
```

Edit the copied script and make the desired change

```
vi scripts/database-qaddb-start.modified
```

Create the patch

```
mkdir patches/scripts
```

```
yab patch-create scripts/database-qaddb-start scripts/database-qaddb-start.modified >  
patches/scripts/database-instance-start.patch
```

```
rm scripts/database-qaddb-start.modified
```

Configuring the patch

Configure the patch to be applied to all files in the scripts directory that match the pattern 'database-*-start' anytime the database-script command is executed.

build/config/configuration.properties

```
patch.database-start.patch=${patches.dir}/scripts/database-instance-start.patch  
patch.database-start.dir=${scripts.dir}  
patch.database-start.includes=database-*-start  
patch.database-start.trigger=database-script
```

Verifying the patch

```
yab database-script
```



Applying a patch may not produce the desired result if the patched file differs greatly from the file that was used to generate the patch.

qra.pl

A patch for the qra.pl procedure library distributed in the qra-oe11 package.

```
> mkdir patches/qra.pl  
> cp [PATCHED qra.pl] patches/qra.pl
```

build/config/configuration.properties

```
# @append  
# @group 3  
propath.base=${patches.dir}/qra.pl/qra.pl
```

Prepared Contexts

An update can be prepared in a context. YAB will consume system resources (CPU, Memory, Disk) while preparing the update, but otherwise the request will have no impact on the live environment.

The ability to prepare an update "in advance", can have the following uses:

- To prepare an update for an environment, while the environment is running, without disrupting users.
- To abandon a failed attempt at preparing an update without having had any impact on the environment.
- To reduce the time the environment is offline when the update is applied, by reducing the work that must be done while the environment is offline.
- To review the changes before they are applied to the environment.
- To have the option to (partially) revert an update that has been applied to the system.

Contexts may also be leveraged as a creative way to periodically (nightly, weekly) validate that an environment is in a suitable state to be updated. For example, an administrator could set up a cron job to prepare the [update](#) command in a context and then delete the context, to validate that the code in an environment compiles.

- [Using a context](#)
- [Adding files to a context](#)
- [Validating a context](#)
- [Logging in a context](#)
- [Context Guidelines](#)
- [Limitations](#)

Using a context

Use the (-c) option to submit a request to run in a context. If the context does not exist it will be created.

Typically you will want to name the context yourself by supplying a meaningful name as the value of the option.

```
> yab -c:test
> yab system-context-list
test  build/contexts/prepared/test
```

But you can omit the name to create a new context with a unique system generated name.

```
> yab -c
65fc
> yab system-context-list
65fc  build/contexts/prepared/65fc
```

In the previous examples, we simply created a context. Our request did not have any commands. Following this, we could submit commands to run in that context by referencing the context by name (-c:NAME), but it is not necessary to create the context in a separate request. For example, either of the following requests will install *i19-es-2.2.0.7* in a context named "install-i19-es".

```
> yab -c:install-i19-es
> yab -c:install-i19-es install i19-es-2.2.0.7.zip
```

or

```
> yab -c:install-i19-es install i19-es-2.2.0.7.zip
```

We might even install multiple products into a context and use the (-install-update:false) option to defer the update until the end.

```
> yab -c:install-i19 -install-update:false install i19-es-2.2.0.7.zip
> yab -c:install-i19 -install-update:false install i19-de-2.2.0.7.zip
> yab -c:install-i19 update
```

Nothing we have done up to this point will have had an effect on the environment. We can see this by comparing the result of running the [info](#) command outside of the context with the result of running it in the context. The info outside of the context will not list our pending i19 products (because they have not been applied) but the info run in the context will. (You can use other YAB commands ([help](#), [config](#)) to query the state of the context as well.)

```
> yab info
> yab -c:install-i19 info
```

Use the explain-apply command to get a detailed view of what would happen if the changes in the context were applied, without actually applying the changes.

```
> yab -c:install-i19 explain-apply
```



Only the changes produced by the **last** request are saved in the context, with the following exceptions:

- Requests that do not introduce changes ([help](#), [info](#), [config](#), [explain-apply](#)) are safe to run at any time.
- Package and configuration changes are retained across requests.

The practical takeaway is that the last command in a context before the context is applied should almost always be `update` or a command that runs `update` like `install`.

When you are ready to apply the changes prepared in the context to the environment, use the `apply` command.

```
> yab -c:install-i19 apply
```

In the typical case where you have prepared an update, the environment will be taken offline for a portion of the time.

If the apply fails, you can revert the changes prepared in the context.

```
> yab -c:install-i19 revert
```

Not all changes that can be applied can be reverted, but usually a `revert` with some manual adjustments, is preferable to having to manually roll back all of the changes that were applied to the environment.

The `system-context-info` command reports on the processes (and optionally operations) that have been applied to the environment.

```
> yab -c:install-i19 system-context-info
```

When you are through with the context, you can delete it to reclaim disk space:

```
> yab system-context-delete install-i19
```

Adding files to a context

A context is associated with a "cache" to store pending file system changes.

`build/contexts/prepared/[NAME]/fs`

When you install software in a context, you do not need to interact directly with the cache. For certain kinds of changes, however, you may need to work with the cache. For example, consider a scenario where you want to prepare a change to `configuration.properties` in a context. To set things up you would use the `system-context-locate` command with the `(-pull)` option to add `configuration.properties` into the context.

```
> yab -c:cfg -pull system-context-locate build/config/configuration.properties  
/dr01/qadapps/ee/build/contexts/prepared/cfg/fs/rw/_/build/config/configuration.properties
```

You could then edit the file in the context.

```
> vi /dr01/qadapps/ee/build/contexts/prepared/cfg/fs/rw/_/build/config/configuration.properties
```

Run an update in the context (as needed).

```
> yab -c:cfg update
```

And then apply the change to the environment.

```
> yab -c:cfg apply
```

Validating a context

Using a context introduces the risk of conflicts.

For example, applying a change prepared in a context typically will restart the environment. Standard operating procedures, may require that you schedule downtime and while waiting for the downtime window, a file with a pending update in the context could be modified in the environment. Applying the change would lose the changes made in the environment. To address this problem, the context includes information describing the original state of the environment, and when the environment is no longer in this state, the request will be flagged as INVALID. The `explain-apply` command can be used at any time to check whether or not a context is valid to be applied (and likewise the `explain-revert` command to check if the context is valid to be reverted). The `apply` command will also perform these validations up front and reject the request if any of the changes are INVALID.



The `(-errors)` option on the `system-context-info` command provides a focused way to see the details behind any integrity violations.

Logging in a context

Log messages are recorded in the context while the context is prepared:

`build/contexts/prepared/[NAME]/logs/yab.log`

When changes are applied to the environment (apply or revert) log messages are recorded in the context and in the environment:

`build/logs/yab.log`

Context Guidelines

A context can be leveraged to prepare a change in advance without impacting the environment, but this separation from the environment increases the difficulty of resolving failures. By following the guidelines below, you can reduce the chance of failure while being prepared to respond if a failure occurs.

Preparation

Verify the environment is in working order

Ensure that all environment components are running:

```
> yab -verify status
```

Ensure that there are no outstanding validation errors:

```
> yab validate
```



If the environment can be taken offline, an update should be run before verifying the environment.

Schedule an administrative quiet period

Users may continue to use the environment while the update is being prepared, but all admin changes, apart from the work in the context, should be suspended from the point at which the context is created until it has been applied to the environment. Performing admin work on the environment (changing configuration settings, running an update), after the context is created and before it is applied, may invalidate the work in the context.

Handling Failures

Before Apply

Until the context is applied to the environment, work in the context has no effect on the environment. The safest strategy for recovering from a failure that occurs before the context is applied, is to determine the underlying cause of the failure, delete the context, and then start over with an appropriately revised plan.

In limited circumstances where this extra precaution is not necessary it may be possible to correct a failure by adjusting the context (changing configuration settings, installing new software), followed by reprocessing the change in the context.

During Apply

Invalid Status

When the apply fails with an INVALID status, this indicates that the environment has changed since the context was created. The context log file will contain detailed information on the files that have changed. These integrity errors are raised during an initial verification before the environment has been modified and can be handled in the same manner as a "Before Apply" failure.

In limited circumstances, the integrity error will describe a change that is not substantive (for example a file timestamp has changed where nothing else is different) and the administrator can choose to force the apply using the (-force) option.



Forcing an apply indiscriminately could result in losing changes to the environment as the "out of date" files in the context are applied to the system.

Other Errors

The system-cache-flush step integrates packages and configuration changes into the environment. If a failure occurs before this step, the error can be handled in the same manner as a "Before Apply" failure.

If a failure occurs after system-cache-flush, you have the choice of trying to continue integrating the prepared change or trying to restore the system to its previous state. Whether it is best to move forward or back depends on a number of factors and is not a decision that is easy to generalize about.

Moving Forward

The recommended approach is to continue by working directly with the environment and not the context. In other words, make the appropriate corrections directly in the environment and then process the changes by running an update in the environment.

In limited circumstances (when you know what you are doing), corrections can be applied to the context and/or environment and the apply can be reissued (it will resume where it previously failed).

Moving Back

Before the context is applied, the system creates a configuration backup with the name "before-CONTEXT" where CONTEXT is the name you assigned to the context. To restore the configuration backup use the (-config-restore) option and then run an update to apply the change.

Ex. Restore the configuration before the "test" context was applied.

```
> yab -config-restore:before-test  
> yab update
```

Limitations

Some upgrades cannot be prepared.

Some applications are sensitive to version differences between client and server processes. OpenEdge, for example, cannot be upgraded in a context, because a shared memory connection from a client in the context (new version) to a server in the environment (old version) could fail.

Some commands cannot be prepared.

It is not possible (or technically infeasible) to prepare certain commands.

Some commands that can be prepared and applied cannot be reverted.

It is not possible (or technically infeasible) to revert certain commands.

Limited Coverage

We can prepare configuration, package, and Progress schema, (XML) data, and code changes. We plan on increasing coverage in subsequent releases.

Apply/Revert are valid for a limited time

Contexts were not designed to store changes for long periods of time. See [Validating a context](#) for an explanation of why this is the case.



You can prepare a request even when some of the steps/commands are not prepared. The commands that cannot be prepared are deferred (status DEFER) until the context is applied to the environment at which point they are run in the proper sequence along with the commands that were prepared in advance.

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.

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. Update the configuration of the MFG application server.

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

The [System Configuration](#) page includes an example process for enabling/disabling 4GL trace on an application server.



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

Financials Logging

To troubleshoot a Financials synchronization error you can enable additional logging of the flow by editing the `Synchronize.config` file and setting the `BLDebugLevel` to 31. This will only impact synchronization and not other Financials sessions.

config/Synchronize.config

```
BLDebugLevel=31
```

Additional details will be logged to a file name `ct<1234567>.log` where 1234567 is the sessionID (random number) used by the process.

The location of the log is configured by the setting `fin.serverxml.logging-directory` which defaults to the standard logs directory:

```
fin.serverxml.logging-directory=${appdir.logs}
```

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
  ...
```

See [system-diagnostics](#)

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. 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

[REFRESH]
-----
packages                                OK (3.139 s)
-----
                                (14 tasks)                                [APPLY]
-----
1/14 system-config-package-init
Press ENTER to run 'system-config-package-init' ('s' skip, 'a' run all, 'n' run all to next
break, 'sn' skip all to next break, 'q' quit).
>

OK (15.915 s)
2/14 system-config-before-configure
Press ENTER to run 'system-config-before-configure' ('s' skip, 'a' run all, 'n' run all to next
break, 'sn' skip all to next break, 'q' quit).
>

OK (4.293 s)
3/14 extension-scan
Press ENTER to run 'extension-scan' ('s' skip, 'a' run all, 'n' run all to next break, 'sn' skip
all to next break, 'q' quit).
>
```

The system can be configured to break (prompt) at a specific set of steps instead of at every step by configuring the option (-break) with a comma separated list of steps to break at.

```
> yab -debug-pause -break:netui-pro-compile,bootstrap-default-update update
```

By default, the steps that precede the first break step will be executed without prompting. Use the (-skip) option to skip the steps that precede the first break step, which may be useful to resume processing at a point of failure.

```
> yab -debug-pause -break:netui-pro-compile,bootstrap-default-update -skip update
```

The prompt options control whether the current step is executed and at what point the system should prompt again.

Key	Description
{ENTER}	Executes the current step and prompts at the next step.
s	Skips the current step and prompts at the next step.
n	Executes the current step and prompts at the next break step.
sn	Skips the current step and prompts at the next break step.
a	Executes the current step and all remaining steps.
q	Stops processing the request.

In the following example an update is requested. The (non-system) steps preceding the first break step "netui-pro-compile" are skipped. When prompted, the option "n" was chosen to execute all steps until the next break step "bootstrap-default-update" at which point "q" was chosen to quit the program.

```

> yab -debug-pause -break:netui-pro-compile,bootstrap-default-update -skip update
[REFRESH]
-----
packages OK (3.398 s)
-----
(14 tasks) [APPLY]
-----
1/14 system-config-package-init OK (0.239 s)
2/14 system-config-before-configure OK (0.011 s)
3/14 extension-scan OK (0.001 s)
4/14 system-config-extension-init OK (0.068 s)
5/14 content-scan OK (1.432 s)
6/14 content-configure OK (0.300 s)
7/14 system-config-configure OK (0.419 s)
8/14 system-config-final-configure OK (0.603 s)
9/14 system-config-validate OK (0.720 s)
10/14 system-config-after-configure OK (0.335 s)
11/14 system-config-final-after-configure OK (0.000 s)
12/14 content-order OK (0.001 s)
13/14 system-config-save OK (0.180 s)
14/14 system-cache-flush OK (0.001 s)
-----
update (599 tasks) [APPLY]
-----
1/599 openedge-upgrade-check SKIPPED (0.000 s)
2/599 config-host-update SKIPPED (0.000 s)
3/599 trans-dotdees-fhd-stage SKIPPED (0.000 s)
...
348/599 netui-language-configscreens-update SKIPPED (0.000 s)
349/599 netui-pro-config-update SKIPPED (0.000 s)
350/599 netui-pro-compile
Press ENTER to run 'netui-pro-compile' ('s' skip, 'a' run all, 'n' run all to next break, 'sn'
skip all to next break, 'q' quit).
> n

OK (1:09 m)
351/599 netui-pro-install OK (13.470 s)
352/599 bootstrap-default-update
Press ENTER to run 'bootstrap-default-update' ('s' skip, 'a' run all, 'n' run all to next break,
'sn' skip all to next break, 'q' quit).
> q

SKIPPED (21.178 s)
-----

BUILD CANCELLED (1:52 m)

```



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

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

1. openedge-upgrade-check
2. config-host-update
3. trans-dotdees-fhd-stage
4. qxtend-stage
- ...

Disabling 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 processes that should not be executed in this environment.
#
# Each process should be listed on a new line and lines beginning with
# the '#' character are handled as comments. Use the refresh option (-r) to
# instruct YAB to reprocess this file.
#
#
# When a request is received to directly execute a process listed in this
# file, the system will raise an error that the process is disabled.
# When a request is received to execute a process that implies a process
# listed in this file, the request will be processed but the process listed
# in this file will not be executed. Adding a process to this file, also
# disables any relationships between the process and other processes. So,
# for example, if process (a) implies process (b) and process (b) implies
# process (c) and (b) is disabled, a request to execute (a) will not execute
# (c) unless it is associated with (a) independent of (b).
#
metadata-qracore-update
```

The process-ignore file is re-processed when the system is [refreshed](#).

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

The system-process-disable and system-process-enable commands can be used as an alternative to editing the process-ignore file directly. The command system-process-disable-list lists the processes that are disabled.

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 ...
```

The [check-for-updates setting](#) controls whether or not a refresh will reevaluate packages.

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
```

Validation Settings

It is occasionally necessary or desirable to treat validation errors as non-critical warnings or to skip specific validations.

Use the following settings to adjust the validation mechanism.

```
> yab help validation

SETTINGS

    validation.error          Whether the build should be halted if the configuration is
    not valid.

                                Default: true

    validation.skip.keys      A comma-delimited list of configuration settings that should
    not be validated.

    validation.skip.validators A comma-delimited list of validators that should be skipped.
    g. int) or their fully    NOTE: Validators are identified by their "friendly" name (e.
    CodePageValidation).      qualified class name (e.g. com.qad.yab.ee.foundation.
```

Ex. Treat all validation errors as warnings.

```
build/config/configuration.properties
```

```
validation.error=false
```

Ex. Skip the validations associated with the 'schema.qrbridge-qadadm.area.map' setting.

```
build/config/configuration.properties
```

```
validation.skip.keys=schema.qrbridge-qadadm.area.map
```

Ex. Skip the codepage validation.

```
build/config/configuration.properties
```

```
validation.skip.validators=com.qad.yab.ee.foundation.CodePageValidation
```



Validation settings cannot be defined as request parameters but must be set in a configuration file.

Defining 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"
```

Adding Packages to Local Catalog

In the course of troubleshooting a problem it might be necessary to add a package directly into the local catalog. Adding a package into the catalog will replace all other versions of the package in the local catalog. For example, adding the package `dde-core-1.0.0.0` will replace all other versions of the `dde-core` package in the local catalog.

Add Package

Ex. Add `dde-core-1.1.0.0` into the local catalog.

```
> yab -i:dde-core-1.1.0.0.zip
```

Remove Package

Ex. Remove `dde-core-1.1.0.0` from the local catalog.

```
> rm -rf build/catalog/packages/dde-core  
> yab -r info
```



Adding a package into the local catalog will add the package to the local catalog but it will not be used by the application unless it is [configured and applied](#).

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.



Only the start and stop commands are certified for parallel execution.

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 -threads: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 -threads: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

```

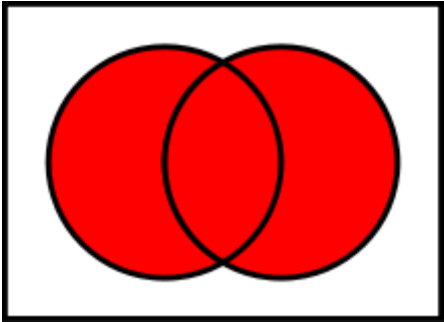
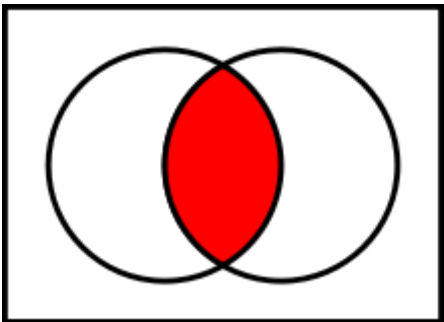
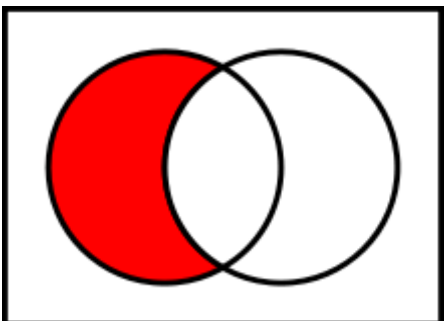
Command Expressions

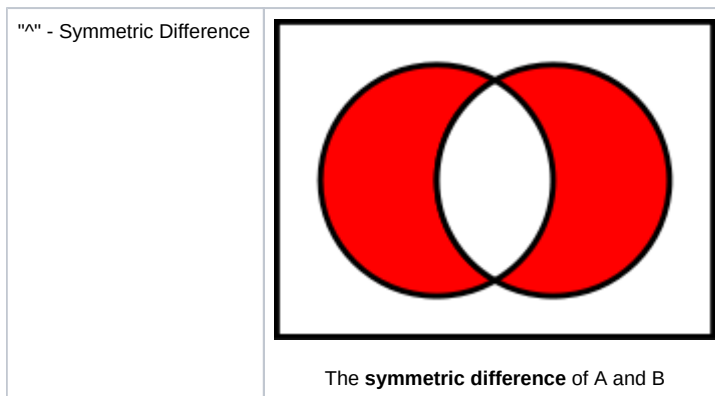
A command expression is a statement that operates on commands as sets.

Syntax

```
<expr> ::= ["(" <command>|<expr> <operator> <command>|<expr>... [")"]
<operator> ::= "|", "&", "~", "^"
<shallow modifier> ::= "!"<command>
```

Operators

Operator	Visual
" " - Union	 The union of A and B
"&" - Intersection	 The intersection of A and B
"~" - Relative Complement	 The relative complement of B in A



By default references to commands are "deep" in that they describe the set formed by the command and its (transitive) implied commands. The shallow modifier changes this to be a reference to the command alone.

Practical Examples

Troublesome Command

Consider a situation in which the environment cannot be cleanly shutdown because the step to stop the admin database raises errors. Everything in the environment can be shutdown while excluding the step to shutdown the admin database with the following request, which demonstrates how command expressions can be used in place of commands when submitting requests.

```
yab "stop-database-qadadm-stop"
```

For completeness, another approach to this problem, that attempts to stop the admin database but ignores failures is the following:

```
yab -failonerror:false stop
```

Alias

A command can be defined that is an alias of another command.

Ex. Define the command "compile" to execute the command "code-mfg-update".

```
process.compile.id=compile
process.compile.expression=code-mfg-update
```

```
> yab compile
```

Selecting Files

Many configuration types use a similar idiom for selecting files defined in terms of a base directory and include and/or exclude patterns to select files from the base directory. For example, the `copy` configuration type is used to configure a file copy. The `source` defines the base directory which contains the files to copy and the `includes` and `excludes` patterns are used to define the files to copy. The pattern matching algorithm is loosely based on the FileSet patterns in the Apache ANT framework, with support for the following wildcard characters:

Wildcard	Description
*	Matches zero or more characters.
?	Matches a single character.
**	Matches zero or more directories.

Ex. Copy all files.

```
copy.test.includes=**/*
```

Ex. Copy all files in the `foo` and `bar` directory.

```
copy.test.includes=foo/*,bar/*
```

Ex. Copy the `foo.css` and `bar.html` files.

```
copy.test.includes=foo.css,bar.html
```

Ex. Copy all XML files except those ending in `"-mapping.xml"`.

```
copy.test.includes=**/*.xml  
copy.test.excludes=**/*-mapping.xml
```

Extensions

An extension is a named directory that is registered with the environment.

An extension is like a [package](#) in that both identify directories containing resources for the environment. However, a package is a versioned, readonly resource that is comparable across systems and time, whereas an extension is an unversioned resource that may change over time.

Add Extension

Extensions may be either explicitly configured or discovered.

Ex. Explicit Configuration

build/config/configuration.properties
<pre>extension.test.name=test extension.test.dir=\${appdir}/test</pre>

Ex. Discovered

Sub-directories in the `extensions.dir` will be processed as extensions when YAB is refreshed (-r), using the sub-directory name to name the extension.

<pre>> yab config extensions.dir extensions.dir=/dr01/qadapps/qa/extensions > mkdir -p extensions/test > yab -r config extension.test.* extension.test.name=test extension.test.dir=/dr01/qadapps/qa/extensions/bar</pre>
--

Discovered extensions can bypass the default extension configuration by defining a file named `extension.properties` that defines the desired configuration.

[EXTENSION DIR]/yab/config/extension.properties

Ex. Discovery with explicit configuration

extensions/test/yab/config/extension.properties
<pre>extension.testa.name=testa extension.testa.dir=a extension.testb.name=testb extension.testb.dir=b</pre>



If the `extension.properties` file does not define the location of an extension ("dir" setting), the directory will default to the discovered extension directory (e.g. `extensions/test`). If an extension is defined with a relative "dir" setting, the location is resolved relative to the discovered extension directory (e.g. `a = extensions/test/a`).

List Extensions

Extensions are listed by the [info](#) command (below packages).

<pre>> yab info ... EXTENSIONS testa testb</pre>	<pre>extensions/test/a extensions/test/b</pre>
--	--

Remove Extension

An extension that is explicitly configured is removed by removing the configuration. Otherwise a discovered extension can be removed by removing the extension directory (or moving it outside of the `extensions.dir`) and then refreshing YAB.

```
> rm -rf extensions/test
> yab -r info
```

Extension Configuration

An extension can distribute configuration settings that will be integrated into the environment. The following table describes the expected location for configuration files of default and important priority and where they are integrated into the system.

Source Files	Destination
[EXTENSION DIR]/yab/config/*.properties	build/config/extensions/[EXTENSION NAME]/default
[EXTENSION DIR]/yab/config/important/*.properties	build/config/extensions/[EXTENSION NAME]/important

The configuration settings contributed by (or on behalf of extensions) have the following precedence in the overall configuration processing (high to low):

```
build/config/configuration.properties
build/config/system/*
build/config/extensions/.../important/*
build/config/packages/.../important/*
build/config/extensions/.../default/*
build/config/packages/.../default/*
```

Discovering Content

By default all extensions are scanned for content to integrate into the environment. The discovery process can be disabled by distributing a file named `content-info.properties` in the extension's root directory with the following setting:

content-info.properties

```
scan.enabled=false
```

Rebuild

Rebuild resets a component to its factory default state.



This operation should not be used in a production environment.

Rebuilding components can result in a loss of data. Rebuilding a database, for example, will restore the database to its original condition and will delete any transaction data entered. Rebuilding a web application, will remove files created in the application.

The following components support a rebuild command:

- Database
- Adminserver
- Nameserver
- Appserver
- AIA
- Code
- Tomcat
- Web application
- Webspeed Server
- Webservices Adapter

A rebuild should be followed by an update to apply necessary changes to the rebuilt component.

Ex Rebuild all databases (and reapply the QXtend default configuration).

```
> yab database-rebuild
> yab update
> yab -qxtend.reconfigure qxo-services-stop qxtend-config-update
```