



Nuts, the Java Package Manager

Version v\${apiVersion}, 2020-09-10

Table of Contents

/	1
Nuts	2
Building	3
Introduction	4
nuts stands for Network Updatable Things Services** tool. It is a simple tool for managing remote	4
nuts is a swiss army knife tool as it acts like (and supports) maven** build tool to have an abstract	4
COMMON VERBS:	4
License	6
Nuts and Maven	7
You'd still be Maven, yet you gonna be Nuts	8
Nuts Package manager	9
nuts ** is actually :	9
Transitive dependency resolution manager	9
nuts ** calculates transitive dependencies of an application to resolve other packages to download at install or	9
Package manager	10
nuts ** uses this dependency resolution to help install, update, remove and search for applications. To be able to use an	10
nuts ** also handles search for newer versions and update the installed application at request. Updating a software does not	10
Feature rich Toolset	11
nuts ** is intended to be used either by human users or by robots and other applications. It comes with portable,	11
nuts ** is mainly a commandline program that helps installing, uninstalling, searching, updating and running artifacts	11
nsh** brings the bash facilities to all environments (windows included) in a very portable manner. Besides it integrates	11
nadmin is intended for configuring nuts ** workspaces, managing repositories and users. It helps also configuring	11
ndi**, is the tool for a seamless integration in your operating system. It mainly configures user PATH variable environment and	11
Nuts Workspaces	11
Application Framework	12

Nuts** can also be embedded as a library in your application. This enables you to wire classes on the fly by its network	12
Nuts ? Really ?	12
Let's start the journey	12
nuts ** is well installed, just restart your terminal.	13
Nuts projects	15
nuts ** repository is composed of several projects that can be organized in 5 categories	15
Core nuts : These projects are the core/base of the nuts package manager	16
Companion Tools : These projects are applications and tools to install with nuts itself. Their installation are prompted at first install of Nuts.	17
Toolbox : These projects are applications and tools built on top of nuts Application Framework and are of common interest	18
Lib : These projects are common libraries that can be used to enable some nuts features in your application	19
Other : All other apps that does not fit in the previous categories	20
1- Core Nuts projects	20
nuts-builder** is a meta project (parent maven pom project) that helps building all the other projects.	20
1.2 nuts-api	20
nuts-api is the effective "Nuts" only required dependency. It defines the bootstrap application that is responsible of loading all necessary libraries for its execution. nuts-api starts to load nuts-core which is responsible of implementing all features and interfaces declared by the nuts-api** library. That implementation will handle further download, version dependency etc. Such architecture is considered to force loose coupling with nuts binaries.....	20
nuts-api** is a very thin bootstrapper : its size is about 300k. It can be used as a standalone application or as an embedded library.	20
1.3 nuts-core	20
nuts-core is the effective and standard implementation of nuts-api. nuts-core has a faster update pace than nuts-api. It focuses on performance and compliance to the Nuts specification declared by nuts-api interfaces. You are not required to add this dependency to your application if you want to embed Nuts**. The library will be loaded on the wire (if not yet present in the classpath).	20
nuts-core is designed to have very few dependencies : gson and jansi. gson trivially is used to support json serialization : the main format used in Nuts to support configuration and descriptors. jansi** is used to support terminal coloring and the "Nuts Text Coloring Format" (NTCF), a simple text format that helps creating colorful terminal applications.	20

2. Companion tools projects	20
nadmin (for nuts admin) is an administration tool to the nuts** workspaces. It adds support to manage users, credentials, authorizations, workspaces and repositories by providing command line support for such actions.	21
2.2 nsh	21
nsh (for nuts** shell) is simply a portable POSIX bash compatible implementation. It supports all common builtin commands (ls, cd, rm, ...) and adds support to grep, ssh and scp in a seamless manner. It also supports command line, scripts (including commons constructs with if, do, case, ...) and pipes ().....	21
2.3 ndi	21
ndi (for nuts desktop integration) is simply a helper tool to support seamless integration of nuts commands in your favorite operating system and environment. ndi is responsible of creating script shortcuts to your common commands so that you can invoke tem directly from your environment. For instance it creates an "nadmin" script and configures your PATH environment to help calling the nuts admin tool instead of the common way to do so "nuts nadmin". On window system, ndi** will create shortcuts and menus.	21
3. Toolbox projects	21
nuts** comes with an array of tools out of the box you can install and play with. Here are these tools.....	21
3.1 nded	21
nded for " nuts Descriptor Editor" is a small tool for creating and editing json nuts** descriptor file. It is intended to be called by automation tools.	21
3.2 ncloon	21
ncloon is an angular web application frontend for nuts** . It helps navigating, searching and installing artifacts. It is intended to be a web admin tool as well.	21
3.3 nservr	21
nservr is a standalone application that runs a small http server that will expose a workspace as a remote repository to other Nuts** installations. This is the simplest way to mirror a workspace and share artifacts between networked nodes.	22
3.4 nwarr	22
nwarr (for nuts Web Application Archive) is a web application that exposes nservr** as a war to be deployed on a more mature http server or container.	22
3.5 ndexer	22
ndexer (for Indexer) is a lucene powered index for nuts . It can be shared across multiple nuts** workspaces and processes.	22
3.6 feenoo	22
feenoo** is a small search tool. It searches for files, files contents and classes within jars.	

You can search for files than contains some text or jars that contain some class, or jars of a specific version of java.....	22
file-version** is a small tool that helps detecting files versions. It supports jar, war, ear, dll and exe file versions. It opens a file and looks for it's version in its meta-data.	22
3.8 nmysql	22
nmysql** is a companion tool to the mysql and mariadb servers. The initial actions supported are backup and restore including push/pull mechanism from/to a couple of databases for synchronization. It supports jdbc and ssh based access to remote mysql/mariadb installation.....	22
3.9 ntomcat	22
ntomcat** is a companion tool to the tomcat http server. The initial actions supported are start, stop, status, configure (http ports etc..) and deploy. It supports as well installation of several versions of Tomcat and multi domain configuration for deployments.	22
3.10 nderby	22
nderby** is a companion tool to the derby database server. The initial actions supported are start, stop, status and configure. It supports as well installation of several versions of Derby.	23
3.11 nmvn	23
nmvn** is a companion tool to maven. It supports installations of several versions of it and running them seamlessly.	23
3.12 worky	23
worky is a developer centered tool. The 'y' in worky refers to 'my' in the "Tunisian Dialect" and hence means "my work". Worky is the tool we - maven users - need to check if the version of project we are working on is yet to be deployed to nexus or not. So basically it checks if the version is the same, and downloads the server's version and then compares binaries to local project's to check if we have missed to update the version in our pom.xml. I know I'm not the only one having pain with jar deployments to nexus. Worky** does other things as well to help me on on daily basis.	23
4. Library Projects	23
6. Honorable mentions	24
Getting Started	25
Command Line Arguments	26
Nuts Commandline	27
Nuts supports a specific format for command line arguments. This format is the format supported in nuts ** Application Framework (NAF) and as such all NAF applications support the same command line arguments format.	27
Option Values	27

Boolean Options	27
Combo Simple Options	27
Ignoring Options, Comments	27
Nuts Option Types	28
Create Options : such options are only relevant when creating a new workspace. They define the configuration of the workspace to create. They will be ignored when the workspace already exists. Examples include	29
Open Options : such options are relevant when creating a new workspace or when opening an existing workspace. They define the way commands are executed. Examples include	30
Exported Options : are passed to sub-nuts-processes that will be created by nuts. For instance when nuts will call the nsh command it will spawn a new process. In such case, these options are passed to the sub-process as environment variable.....	31
Application Options : are options that are by default supported by Applications using NAF (Nuts Application Framework) (as well as Nuts it self).	32
Frequently Asked Questions	33
Why do we need a package manager for Java. Isn't Maven enough?	33
nuts** stands for "Network Updatable Things Services". It helps managing things (artifacts of any type, not only java).	33
Does nuts support only jar packaging.....	33
Can I contribute to the project	33
Where can I find Documentation about the Project.....	34
How can I make my application "Nuts aware"	34
Why should I consider implementing my terminal application using Nuts Application Framework (NAF)	34
seamless integration with nuts and all other NAF applications	35
support standard file system layout (XDG) where config files and log files are not necessarily in the same folder see [Nuts File System](./advanced/filesystem.md) for more details.	36
support application life cycle events (onInstall, onUninstall, onUpgrade),	37
standard support of command line arguments	38
dynamic dependency aware class loading.....	39
terminal coloring and components (progress bar, etc...)	40
json,xml,table,tree and plain format support out of the box	41
pipe manipulation when calling sub processes	42
advanced io features (persistence Locks, monitored downloads, compression, file hashing....) ..	43
standard ways to support and use installed platforms (installed JRE, JDK, ...)	44
and more...	45
Can I use NAF for non terminal applications, Swing or JavaFX perhaps	45

What is the License used in Nuts	45
nuts ** is published under GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later.....	45
Installation	46
System Requirements	46
Installation	46
: this shortcut will open a configured command terminal. nuts command will be available as well as several nuts companion tools installed by ndi by default.....	48
Test Installation	49
Run a command	50
Running a deployed artifact	50
Artifact Long Ids	50
nuts ** long ids are a string representation of a unique identifier of the artifact. It has the following form :	50
Artifact Installation	51
Multiple Artifact version Installation	51
Searching artifacts	52
Running local jar file with its dependencies	53
Application Framework	54
Nuts Applications	55
Your first Nuts Application	56
API Reference	57
Application	58
☕ NutsApplication	58
⌚⌚ Static Methods	61
⚙ Instance Methods	61
☕ NutsApplicationContext	63
⌚ Instance Fields	64
⌚ Instance Properties	64
⚙ Instance Methods	69
☕ NutsApplicationLifeCycle	73
☕ NutsApplications	74
⌚⌚⌚ Constant Fields	74
⌚⌚⌚ Static Properties	74
⌚⌚⚙ Static Methods	75
⌚ Constructors	75

Base	76
☕ Nuts	76
🔇 Static Properties	76
🔇⚙️ Static Methods	76
⌚ Instance Properties	83
⚙️ Instance Methods	86
〝 NutsIOCompressAction	86
⌚ Instance Properties	86
⚙️ Instance Methods	88
☕ NutsIdLocationBuilder	93
⚙️ Instance Methods	94
☕ NutsIndexStore	95
⌚ Instance Properties	95
⚙️ Instance Methods	96
〝 NutsIndexStoreFactory	97
⚙️ Instance Methods	98
〝 NutsInputStreamTransparentAdapter	98
⚙️ Instance Methods	98
☕ NutsInstallCommand	98
⌚ Instance Properties	99
⚙️ Instance Methods	100
〝 NutsInstallEvent	106
⌚ Instance Properties	106
〝 NutsInstallInformation	107
⌚ Instance Properties	107
☕ NutsListener	109
〝 NutsMapListener	109
⚙️ Instance Methods	110
〝 NutsProcessInfo	110
⌚ Instance Properties	110
〝 NutsProgressFactory	111
〝 NutsProgressMonitor	112
⚙️ Instance Methods	112
〝 NutsQuestion	113
⌚ Instance Properties	113
⚙️ Instance Methods	115

☕ NutsQuestionParser	119
⚙ Instance Methods	119
☕ NutsQuestionValidator	119
⚙ Instance Methods	119
☕ NutsRepository	119
⌚ Instance Fields	120
⌚ Instance Properties	120
⚙ Instance Methods	122
☕ NutsRepositoryFilter	126
⚙ Instance Methods	126
☕ NutsResultList	126
⚙ Instance Methods	126
☕ NutsSearchId	128
⚙ Instance Methods	128
☕ NutsSearchIdFilter	129
⚙ Instance Methods	129
☕ NutsSession	129
⌚ Instance Properties	129
⚙ Instance Methods	135
☕ NutsSessionTerminal	142
⌚ Instance Properties	142
⚙ Instance Methods	143
☕ NutsTokenFilter	143
⌚ Instance Properties	143
⚙ Instance Methods	144
☕ NutsVersionInterval	144
⌚ Instance Properties	144
⚙ Instance Methods	145
☕ NutsWorkspace	145
⌚ Instance Properties	146
⚙ Instance Methods	147
Command Line	154
☕ NutsArgument	154
⌚ Instance Properties	154
⚙ Instance Methods	159
☕ NutsArgumentCandidate	161

Instance Properties	161
NutsArgumentName	161
Instance Properties	161
Instance Methods	162
NutsCommandAutoCompleteBase	162
Instance Properties	162
Instance Methods	163
NutsCommandExecOptions	163
Constant Fields	163
Instance Properties	164
Instance Methods	165
NutsCommandLine	165
Instance Properties	167
Instance Methods	168
NutsCommandLineFormat	176
Instance Properties	176
Instance Methods	177
NutsCommandLineProcessor	179
Instance Methods	179
NutsConfigurable	181
Instance Methods	181
Commands	183
NutsDeployCommand	183
Instance Properties	183
Instance Methods	185
NutsExecCommand	187
Instance Properties	188
Instance Methods	191
NutsFetchCommand	196
Instance Properties	196
Instance Methods	200
NutsUndeployCommand	237
Instance Properties	237
Instance Methods	238
NutsUninstallCommand	239
Instance Properties	239

❶ Instance Methods	240
❷ NutsUpdateOptions	243
❸ Constant Fields	243
❹ Instance Properties	243
❺ NutsUpdateResult	243
❻ Instance Properties	244
❻ NutsWorkspaceCommand	245
❼ Instance Properties	246
❽ Instance Methods	246
❾ NutsWorkspaceUpdateResult	246
❼ Instance Properties	247
Config	249
❷ NutsAddOptions	249
❸ Constant Fields	249
❼ Instance Properties	249
❽ Instance Methods	249
❾ NutsAddRepositoryOptions	250
❸ Constant Fields	250
⠁ Constructors	250
❼ Instance Properties	251
❽ Instance Methods	253
❷ NutsCommandAliasConfig	254
❸ Constant Fields	254
❼ Instance Properties	254
❽ Instance Methods	256
❷ NutsConfigItem	256
❸ Constant Fields	256
❼ Instance Properties	257
❷ NutsDefaultWorkspaceOptions	257
❸ Constant Fields	257
❻ Static Methods	257
⠁ Constructors	258
❼ Instance Properties	258
❽ Instance Methods	271
❷ NutsExtensionInformation	273
❼ Instance Properties	273

` NutsRepositoryConfig	274
` Constant Fields	274
` Constructors	274
` Instance Properties	275
` Instance Methods	277
` NutsRepositoryConfigManager	278
` Instance Properties	278
` Instance Methods	280
` NutsRepositoryDefinition	285
` Constant Fields	285
` Constructors	286
` Instance Properties	286
` Instance Methods	288
` NutsSdkLocation	289
` Constant Fields	289
` Constructors	289
` Instance Properties	289
` Instance Methods	291
` NutsUpdateStatisticsCommand	291
` Instance Properties	292
` Instance Methods	292
` NutsUser	295
` Instance Properties	295
` Instance Methods	296
` NutsUserConfig	296
` Constant Fields	296
` Constructors	296
` Instance Properties	297
` Instance Methods	298
` NutsWorkspaceCommandAlias	299
` Instance Properties	299
` Instance Methods	300
` NutsWorkspaceCommandFactory	300
` Instance Properties	300
` Instance Methods	301
` NutsWorkspaceConfigManager	301

□ Instance Properties	301
⚙ Instance Methods	307
📝 NutsWorkspaceListConfig	319
⌚⌚ Constant Fields	319
∅ Constructors	319
□ Instance Properties	319
⚙ Instance Methods	320
📝 NutsWorkspaceListManager	320
□ Instance Properties	320
⚙ Instance Methods	321
📝 NutsWorkspaceLocation	322
⌚⌚ Constant Fields	322
∅ Constructors	322
□ Instance Properties	322
⚙ Instance Methods	323
📝 NutsWorkspaceOptions	324
∅ Instance Properties	324
option-type :** exported (inherited in child	324
option-type :** runtime (available only for the current workspace instance)	325
option-type :** create (used when creating new workspace. will not be	325
option-type :** runtime (available only for the current workspace instance)	325
option-type :** exported (inherited in child workspaces)	325
option-type :** runtime (available only for the current workspace instance)	326
option-type :** exported (inherited in child workspaces)	326
option-type :** runtime (available only for the current workspace instance)	326
option-type :** exported (inherited in child workspaces)	327
option-type :** runtime (available only for the current workspace instance)	327
option-type :** runtime (available only for the current workspace instance)	327
option-type :** exported (inherited in child workspaces)	327
option-type :** exported (inherited in child workspaces)	328
option-type :** runtime (available only for the current workspace instance)	328
option-type :** runtime (available only for the current workspace instance)	328
option-type :** runtime (available only for the current workspace instance)	328
option-type :** exported (inherited in child workspaces)	329
option-type :** exported (inherited in child workspaces)	329
option-type :** exported (inherited in child workspaces)	329

option-type :** create (used when creating new workspace. will not be	329
option-type :** exported (inherited in child workspaces)	330
option-type :** runtime (available only for the current workspace instance)	330
option-type :** exported (inherited in child workspaces)	330
option-type :** exported (inherited in child workspaces)	330
option-type :** exported (inherited in child workspaces)	331
option-type :** exported (inherited in child	331
option-type :** runtime (available only for the current workspace instance)	331
option-type :** exported (inherited in child workspaces)	331
option-type :** exported (inherited in child workspaces)	332
option-type :** exported (inherited in child workspaces)	332
option-type :** exported (inherited in child workspaces)	332
option-type :** runtime (available only for the current workspace instance)	333
option-type :** create (used when creating new workspace. will not be	333
option-type :** runtime (available only for the current workspace instance)	333
option-type :** exported (inherited in child workspaces)	333
option-type :** runtime (available only for the current workspace instance)	334
option-type :** exported (inherited in child workspaces)	334
option-type :** exported (inherited in child workspaces)	334
option-type :** runtime (available only for the current workspace instance)	335
option-type :** runtime (available only for the current workspace instance)	335
option-type :** runtime (available only for the current workspace instance)	335
option-type :** create (used when creating new workspace. will not be	335
option-type :** create (used when creating new workspace. will not be	336
option-type :** create (used when creating new workspace. will not be	336
option-type :** exported (inherited in child workspaces)	336
option-type :** exported (inherited in child workspaces)	337
option-type :** exported (inherited in child workspaces)	337
option-type :** exported (inherited in child workspaces)	337
option-type :** exported (inherited in child workspaces)	337
option-type :** exported (inherited in child	338
⚙ Instance Methods	338
option-type :** create (used when creating new workspace. will not be	338
option-type :** create (used when creating new workspace. will not be	339
☕ NutsWorkspaceOptionsBuilder	339
✉ Instance Properties	339

option-type :** create (used when creating new workspace. will not be	342
option-type :** runtime (available only for the current workspace instance).....	345
option-type :** create (used when creating new workspace. will not be	346
⚙ Instance Methods	347
☕ NutsWorkspaceStoredConfig	348
⠁ Instance Properties	348
⚙ Instance Methods	350
Constants	351
☕ NutsConstants	351
⠁ Constructors	351
Descriptor	352
☕ NutsClassifierMapping	352
⠁ Instance Properties	352
☕ NutsClassifierMappingBuilder	353
⠁ Instance Properties	353
⚙ Instance Methods	355
☕ NutsContent	356
⠁ Instance Properties	356
☕ NutsDefaultContent	356
⠁ Instance Fields	357
⠁ Constructors	357
⠁ Instance Properties	357
⚙ Instance Methods	358
☕ NutsDependency	358
⚙ Instance Methods	361
☕ NutsDependencyBuilder	361
⠁ Instance Properties	362
⚙ Instance Methods	365
☕ NutsDependencyFilter	366
⚙ Instance Methods	366
☕ NutsDependencyTreeNode	366
⠁ Instance Properties	367
☕ NutsDescriptor	367
⚙ Instance Methods	371
☕ NutsDescriptorBuilder	372
⚙ Instance Methods	376

☕ NutsDescriptorFilter	387
⚙ Instance Methods	387
☕ NutsExecutableInformation	387
⌚ Instance Properties	388
☕ NutsExecutionEntry	389
⌚ Instance Properties	389
☕ NutsId	390
⌚ Instance Properties	390
Events	406
☕ NutsInstallListener	406
Exception	412
☕ NutsAlreadyDeployedException	412
⌚ Constructors	412
☕ NutsAlreadyInstalledException	413
⌚ Constructors	413
☕ NutsElementNotFoundException	414
⌚ Constructors	414
☕ NutsException	416
⌚ Constructors	416
⌚ Instance Properties	418
☕ NutsExecutionException	418
⌚⌚⌚ Constant Fields	419
⌚ Constructors	419
⌚ Instance Properties	420
☕ NutsExtensionAlreadyRegisteredException	421
⌚ Constructors	421
⌚ Instance Properties	421
☕ NutsExtensionException	422
⌚ Constructors	422
⌚ Instance Properties	422
☕ NutsExtensionNotFoundException	423
⌚ Constructors	423
⌚ Instance Properties	423
☕ NutsFactoryException	424
⌚ Constructors	424
☕ NutsFetchModeNotSupportedException	425

 Constructors	425
 Instance Properties	426
 NutsIllegalArgumentException	427
 Constructors	427
 Instance Properties	428
 NutsInstallException	428
 Constructors	428
 NutsInstallationException	429
 Constructors	430
 Instance Properties	430
 NutsInvalidRepositoryException	430
 Constructors	431
 NutsInvalidWorkspaceException	431
 Constructors	431
 Instance Properties	431
 NutsLockAcquireException	432
 Constructors	432
 NutsLockException	433
 Constructors	433
 Instance Properties	434
 NutsLoginException	434
 Constructors	434
 NutsNotExecutableException	435
 Constructors	436
 Instance Properties	436
 NutsNotFoundException	436
 Constructors	437
 Instance Properties	438
 NutsNotInstallableException	438
 Constructors	438
 NutsNotInstalledException	439
 Constructors	439
 NutsParseException	440
 Constructors	441
 Instance Properties	441
 NutsParseEnumException	442
 Constructors	442

⌚ NutsPushException	443
⌚ Constructors	443
⌚ Instance Properties	444
⌚ NutsReadOnlyException	445
⌚ Constructors	445
⌚ NutsRepositoryAlreadyRegisteredException	445
⌚ Constructors	446
⌚ NutsRepositoryException	446
⌚ Constructors	446
⌚ Instance Properties	447
⌚ NutsRepositoryNotFoundException	447
⌚ Constructors	447
⌚ NutsSecurityException	447
⌚ Constructors	448
⌚ Instance Properties	449
⌚ NutsTooManyElementsException	449
⌚ Constructors	449
⌚ NutsUnexpectedException	451
⌚ Constructors	451
⌚ NutsUninstallException	452
⌚ Constructors	452
⌚ NutsUnsatisfiedRequirementsException	453
⌚ Constructors	453
⌚ NutsUnsupportedArgumentException	455
⌚ Constructors	455
⌚ NutsUnsupportedEnumException	456
⌚ Constructors	456
⌚ Instance Properties	457
⌚ NutsUnsupportedOperationException	457
⌚ Constructors	458
⌚ NutsUserCancelException	459
⌚⌚ Constant Fields	459
⌚ Constructors	459
⌚ NutsValidationException	460
⌚ Constructors	460
⌚ NutsWorkspaceAlreadyExistsException	461

⠁ Constructors	462
⠁ Instance Properties	462
☕ NutsWorkspaceException	462
⠁ Constructors	463
☕ NutsWorkspaceNotFoundException	463
⠁ Constructors	463
⠁ Instance Properties	463
Extensions	465
☕ NutsRepositoryModel	465
⠁ Instance Fields	465
⠁ Instance Properties	466
⚙️ Instance Methods	467
☕ NutsWorkspaceExtension	469
⠁ Instance Properties	469
☕ NutsWorkspaceExtensionManager	469
⠁ Instance Properties	469
⚙️ Instance Methods	470
Format	474
☕ NutsDependencyFormat	474
⠁ Instance Properties	474
⚙️ Instance Methods	477
☕ NutsDescriptorFormat	479
⠁ Instance Properties	479
⚙️ Instance Methods	479
☕ NutsElementFormat	482
⠁ Instance Properties	482
⚙️ Instance Methods	483
☕ NutsIdFormat	490
⠁ Instance Properties	490
⚙️ Instance Methods	493
☕ NutsIterableFormat	499
⠁ Instance Properties	499
⚙️ Instance Methods	500
☕ NutsIterableOutput	501
⠁ Instance Properties	501
⚙️ Instance Methods	501

☕ NutsJsonFormat	503
∅ Instance Properties	503
⚙ Instance Methods	503
☕ NutsQuestionFormat	514
⚙ Instance Methods	514
☕ NutsString	514
∅ Constructors	515
∅ Instance Properties	515
⚙ Instance Methods	515
☕ NutsStringFormat	516
∅ Instance Properties	516
⚙ Instance Methods	517
Input Output	540
☕ NutsIOPrintAction	540
∅ Instance Properties	540
⚙ Instance Methods	543
☕ NutsIOWriteAction	547
∅ Instance Properties	548
⚙ Instance Methods	548
☕ NutsIOWriteAction	549
∅ Instance Properties	549
⚙ Instance Methods	550
☕ NutsIOLockAction	552
∅ Instance Properties	552
⚙ Instance Methods	553
☕ NutsIOManager	554
∅ Instance Properties	554
⚙ Instance Methods	555
☕ NutsLock	561
☕ NutsNonFormattedPrintStream	561
☕ NutsOutputStreamTransparentAdapter	561
⚙ Instance Methods	562
☕ NutsSystemTerminal	562
⚙ Instance Methods	562
☕ NutsTerminal	562
∅ Instance Properties	563

⚙ Instance Methods	563
Internal	566
☕ PrivateNutsArgumentsParser	566
🔊⚙ Static Methods	566
∅ Constructors	567
☕ PrivateNutsBootConfigLoader	568
🔊⚙ Static Methods	568
☕ PrivateNutsBootWorkspaceFactoryComparator	569
∅ Instance Fields	569
∅ Constructors	570
⚙ Instance Methods	570
☕ PrivateNutsCommandLine	570
🔊∅ Constant Fields	571
🔊⚙ Static Methods	571
∅ Instance Fields	572
∅ Constructors	573
∅ Instance Properties	573
⚙ Instance Methods	575
☕ PrivateNutsId	582
🔊⚙ Static Methods	582
∅ Constructors	583
∅ Instance Properties	583
⚙ Instance Methods	584
☕ PrivateNutsJsonParser	584
🔊⚙ Static Methods	584
∅ Instance Fields	585
∅ Constructors	585
⚙ Instance Methods	585
☕ PrivateNutsLog	586
🔊∅ Constant Fields	586
∅ Instance Properties	588
⚙ Instance Methods	588
☕ PrivateNutsPlatformUtils	589
🔊∅ Static Properties	589
🔊⚙ Static Methods	589
☕ PrivateNutsTokenFilter	590

` Instance Fields	590
` Constructors	590
` Instance Properties	590
` Instance Methods	591
` PrivateNutsUtils	591
` Constant Fields	591
` Static Methods	592
` PrivateNutsWorkspacesInitInformation	600
` Instance Fields	600
` Instance Properties	600
` Instance Methods	606
Logging	608
` NutsLogConfig	608
` Constant Fields	608
` Constructors	608
` Instance Properties	608
` Instance Methods	610
` NutsLogManager	610
` Instance Properties	610
` Instance Methods	611
` NutsLogger	613
` Instance Methods	613
` NutsLoggerOp	615
` Instance Methods	615
Other	618
` ArgEntry	618
` Instance Properties	618
` ArgumentImpl	618
` Constructors	619
` Instance Properties	619
` Instance Methods	622
` BootstrapURLs	624
` Constant Fields	624
` Constructors	625
` ConfirmDelete	625
` Instance Properties	625

⚙ Instance Methods	625
☕ Deps	626
⌚ Instance Fields	626
⌚ EnvEntry	626
⌚ Instance Properties	626
⌚ Files	627
⌚⌚⌚ Constant Fields	627
⌚ Constructors	628
⌚ Folders	628
⌚⌚⌚ Constant Fields	628
⌚ Constructors	629
⌚ IdProperties	629
⌚⌚⌚ Constant Fields	629
⌚ Constructors	631
⌚ Ids	631
⌚⌚⌚ Constant Fields	631
⌚ Constructors	632
⌚ Mvn	632
⌚⌚⚙ Static Methods	632
⌚ Names	634
⌚⌚⌚ Constant Fields	634
⌚ Constructors	634
⌚ NutsApplicationLifeCycleImpl	634
⌚ Instance Fields	635
⌚ Constructors	635
⚙ Instance Methods	635
⌚ NutsArtifactCallBuilder	636
⚙ Instance Methods	637
⌚ NutsCommandAutoComplete	638
⌚ Instance Properties	638
⚙ Instance Methods	639
⌚ NutsIOPrintValidationException	640
⌚ Constructors	640
⌚ NutsIOPrintValidator	641
⚙ Instance Methods	641
⌚ NutsIOPrintProcessAction	641

Instance Properties	642
Instance Methods	642
` NutsIOWUncompressAction	643
Instance Properties	643
Instance Methods	645
` NutsInfoFormat	650
Instance Properties	650
Instance Methods	651
` NutsLockBarrierException	652
Constructors	652
` NutsLockReleaseException	653
Constructors	653
` NutsMonitorAction	654
Instance Properties	654
Instance Methods	656
` NutsProgressEvent	659
Instance Properties	659
` NutsRepositoryRef	661
` Constant Fields	661
Constructors	661
Instance Properties	662
Instance Methods	663
` NutsRepositorySecurityManager	664
Instance Methods	664
` NutsUpdateCommand	666
Instance Properties	667
Instance Methods	669
` Permissions	676
` Constant Fields	677
Constructors	679
` QueryFaces	679
` Constant Fields	679
Constructors	680
` RepoTypes	680
` Constant Fields	680
Constructors	680

☕ SimpleConfirmDelete	681
∅ Instance Fields	681
∅ Instance Properties	681
⚙ Instance Methods	681
☕ Users	682
∅∅∅ Constant Fields	682
∅ Constructors	682
☕ Versions	682
∅∅∅ Constant Fields	683
∅ Constructors	683
Security	684
☕ NutsAddUserCommand	684
∅ Instance Properties	684
⚙ Instance Methods	685
☕ NutsAuthenticationAgent	688
∅ Instance Properties	688
⚙ Instance Methods	689
☕ NutsRemoveUserCommand	690
∅ Instance Properties	690
⚙ Instance Methods	691
☕ NutsUpdateUserCommand	691
∅ Instance Properties	692
⚙ Instance Methods	694
☕ NutsWorkspaceSecurityManager	700
∅ Instance Properties	700
⚙ Instance Methods	701
SPI Base	706
☕ NutsBootClassLoader	706
∅ Constructors	706
☕ NutsBootWorkspace	706
∅∅∅ Constant Fields	706
∅ Instance Fields	707
∅ Constructors	709
∅ Instance Properties	709
⚙ Instance Methods	710
☕ NutsBootWorkspaceFactory	714

⚙ Instance Methods	714
⌚ NutsCommandAliasFactoryConfig	715
⌚ Constant Fields	715
⌚ Instance Properties	715
⚙ Instance Methods	716
⌚ NutsComponent	717
⌚ Instance Fields	717
⚙ Instance Methods	717
⌚ NutsDefaultSupportLevelContext	718
⌚ Instance Fields	718
⌚ Constructors	718
⌚ Instance Properties	718
⚙ Instance Methods	719
⌚ NutsDeployRepositoryCommand	719
⌚ Instance Properties	719
⚙ Instance Methods	720
⌚ NutsDescriptorContentParserComponent	721
⚙ Instance Methods	721
⌚ NutsDescriptorContentParserContext	721
⌚ NutsExecutorComponent	723
⌚ Instance Properties	723
⚙ Instance Methods	723
⌚ NutsFetchContentRepositoryCommand	724
⌚ Instance Properties	724
⚙ Instance Methods	725
⌚ NutsFetchDescriptorRepositoryCommand	726
⌚ Instance Properties	726
⚙ Instance Methods	727
⌚ NutsInstallerComponent	727
⚙ Instance Methods	727
⌚ NutsPushRepositoryCommand	728
⌚ Instance Properties	728
⚙ Instance Methods	729
⌚ NutsRepositoryCommand	729
⌚ Instance Properties	730
⚙ Instance Methods	730

☕ NutsRepositoryFactoryComponent	730
⚙ Instance Methods	730
☕ NutsRepositoryUndeployCommand	731
⌚ Instance Properties	731
⚙ Instance Methods	732
☕ NutsSearchRepositoryCommand	732
⌚ Instance Properties	732
⚙ Instance Methods	733
☕ NutsSearchVersionsRepositoryCommand	733
⌚ Instance Properties	734
⚙ Instance Methods	735
☕ NutsServiceLoader	735
⚙ Instance Methods	735
☕ NutsSessionTerminalBase	735
☕ NutsSupportLevelContext	736
⌚ Instance Properties	736
☕ NutsSystemTerminalBase	736
☕ NutsTerminalBase	736
⌚ Instance Properties	736
⚙ Instance Methods	738
☕ NutsTransportComponent	738
⚙ Instance Methods	738
☕ NutsTransportConnection	738
⌚ Instance Properties	739
⚙ Instance Methods	739
☕ NutsTransportParamBinaryFilePart	739
⌚ Constructors	739
⌚ Instance Properties	740
⚙ Instance Methods	740
☕ NutsTransportParamBinaryStreamPart	741
⌚ Constructors	741
⌚ Instance Properties	741
☕ NutsTransportParamParamPart	742
⌚ Constructors	742
⌚ Instance Properties	742
⚙ Instance Methods	743

☕ NutsTransportParamPart	743
☕ NutsTransportParamTextFilePart	743
∅ Constructors	744
∅ Instance Properties	744
⚙ Instance Methods	744
☕ NutsTransportParamTextReaderPart	745
∅ Constructors	745
∅ Instance Properties	745
⚙ Instance Methods	746
☕ NutsURLHeader	746
∅ Instance Properties	747
☕ NutsUpdateRepositoryStatisticsCommand	748
∅ Instance Properties	748
⚙ Instance Methods	748
☕ NutsWorkspaceArchetypeComponent	748
∅ Instance Properties	749
⚙ Instance Methods	749
☕ NutsWorkspaceAware	749
☕ NutsWorkspacelInitInformation	749
∅ Instance Properties	750
⚙ Instance Methods	753

/

Nuts

Building

```
  /\ \ \ \ _ _/ / _____  
  / \ \ / / / / _/ ___/  
  / / \ / / / / /(_ _ )  
 \_ \ \ \_,/_\_/ ___/  version vfile-version: unsupported file :  
 /home/vpc/.config/nuts/nuts  
Execution Failed with code 2
```

Introduction

```

  _ _ _ _ _ / / _____
 / \ \ / / / / _/ _/
 / \ / / / / /(_ _ )
 \_ \ \_ , / \_ / ____/   version vfile-version: unsupported file :
 /home/vpc/.config/nuts/nuts
 Execution Failed with code 2

```

nuts stands for Network Updatable Things Services** tool. It is a simple tool for managing remote artifacts, installing these artifacts to the current machine and executing such artifacts on need. Each managed package is also called a **nuts** which is a Network Updatable Thing Service . === **nuts** artifacts are stored into repositories. A repository may be local for storing local **nuts** or remote for accessing remote artifacts (good examples are remote maven repositories). It may also be a proxy repository so that remote artifacts are fetched and cached locally to save network resources. One manages a set of repositories called a workspace. Managed **nuts** (artifacts) have descriptorsthat depicts dependencies between them. This dependency is seamlessly handled by **nuts** ** (tool) to resolve and download on-need dependencies over the wire.

nuts is a swiss army knife tool as it acts like (and supports) maven** build tool to have an abstract view of the the artifacts dependency and like npm, pip or zypper/apt-get package manager tools to install and uninstall artifacts allowing multiple versions of the very same artifact to be installed.

COMMON VERBS:

- **exec** : execute an artifact or a command
- **install** , **uninstall** : install/uninstall an artifact (using its fetched/deployed installer)
- **deploy** , **undeploy** : manage artifacts (artifact installers) on the local repositories
- **update** : update an artifact (using its fetched/deployed installer)
- **fetch** , **push** : download from, upload to remote repositories
- **search** : search for existing/installable artifacts

- **welcome** : a command that does nothing but bootstrapping **nuts** and showing a welcome message.

License

```
  \_\_\_ / /_____
 / \_\_ / / / \_\_/
 / \_\_ / / / \_\_(_\_)_
 \_\_\_ / \_\_ /____/    version vfile-version: unsupported file :
 /home/vpc/.config/nuts/nuts
 Execution Failed with code 2
```

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Copyright © 2016-2020 thevpc

Nuts and Maven

```
  /\ \ \ \ _ _/_ /_ _ _ _  
 / \ \ / / / / _/_/_/_/  
 / / \ / / / / /(_ _ )  
 \_\ \ \_\_,/\_\/_/_/_/  version vfile-version: unsupported file :  
 /home/vpc/.config/nuts/nuts  
Execution Failed with code 2
```

You'd still be Maven, yet you gonna be Nuts

Is there any package manager for Java™ applications? You can google for it and you will find that many have queried this on blogs and forums. In most cases responses point to maven and gradle, the tremendous build tools. However, both maven and gradle are build tools, while helping build packages they lack of deployment features. They bundle every dependency in every package (think of wars, ears and standalone jars). They do not handle installation or upgrading. Apache ivy, as well, while competing with maven build tool does not provide more than transitive dependency management. The main idea behind a package manager is the automation of installation, update, configuration and removal of programs or libraries in a coherent manner with the help of a database that manages binaries and metadata. maven, to consider one, sticks to the build process, an goes no further.

You may also ask, "Why ever, do we need a package manager for Java™ applications". Okkay, let's take some example of Java™ applications. How can we install apache netbeans IDE ? The proper way is to browse to the editor's website, select the proper mirror if applicable, download the archive, uncompress it, chmod the main binary (i'm a linux guy) and adjust PATH environment variable to point to this binary; Quite a pain. What do we do to update it now? Hopefully, the IDE has a solid plugin architecture and an in-app update/upgrade tool that will help the process (in a gui manner of course). The same applies to eclipse and apache tomcat with the exception that apache tomcat does not even bundle an in-app update tool. The same applies too when dealing with other operating systems (Windows, MacOS, ...). Managing Java™ applications is far from helpful.

Furthermore, as Java™ applications are (usually) not bundled in OS-aware installers, you will end up with a spaghetti home directory with applications installed all over your partitions, which - simply - does not mix up with all the work OS-developers have done to separate logs from data, from temporary files, from binaries, etc. Each application will handle it's files in a very specific manner that would make it hard to manage own's disk (automatic archive/backup/restore) or roaming applications across machines, etc.

Moreover, in a world of containers and devops, deployments of Java™ applications need to be automatable and reproducible with the highest level of simplicity, configurability and integrability. Installing tomcat on a custom port should not not be as painful as using a custom Docker image or a complicated Dockerfile or even a custom apache tomcat bundle.

When we recall that Java™ is the one language that has the more versatile number of libraries, frameworks and tools, I find it annoying not to have a decent package manager to make the leap

and provide facilities I find prime in other languages and platforms (`pip` /python, `npm` /nodejs/javascript) and most of linux distribution (`zypper` /opensuse, `dnf` /redhat `apt-get` /debian/ubuntu) Hence I'm introducing here a humble attempt to provide a tiny (300ko) yet powerful package manager for Java™ applications (but not only) that should handle jar files seamlessly (with little or no modification) and that comes with a set of portable tools that makes this management at a higher level. I'm not talking about redefining the wheel. I'm aware that many tools such as maven, are already very good at what they do, I just needed to make the leap for deployments. You will be able to deploy your applications without bundling all of their dependencies : `nuts` will take care of that. So you'd still be maven, yet you gonna be `nuts`.

Nuts Package manager

`nuts`** is actually :

- a transitive dependency resolution manager
- package manager (backports maven and supports maven repositories)
- automation tool
- feature rich toolset
- application framework
- decentralized
- sandbox based

Transitive dependency resolution manager

`nuts`** calculates transitive dependencies of an application to resolve other packages to download at install or

update/upgrade time. So typically, deployed applications should no more bundle their dependencies within the deployed archive. Thus we avoid the annoying fat jars (using maven plugins like 'maven-assembly-plugin' and 'maven-shade-plugin') and lib folders (using 'maven-dependency-plugin'). It will also reuse dependencies and packages across multiple installed applications and hence save disk space, and network bandwidth.

All what `nuts` needs is a descriptor file within the jar file that defines the immediate dependencies. It then calculates all transitive dependencies automatically. And guess what, all

maven built jars already contain that descriptor : the pom.xml file. So basically all maven applications are already **nuts** aware applications.

Package manager

nuts** uses this dependency resolution to help install, update, remove and search for applications.
To be able to use an

application, it has to be installed and configured with all of its dependencies. This is the main goal of **nuts**. When we ask to install tomcat, for instance, it will search for the best version in registered repositories, download it, and configure it to be ready for execution. The best version is not always the latest one. Actually it would be the latest valid one, thus the latest one that matches some constraints. Constraints include the version of the running java (tomcat 8 works on java 7 but not 6 for instance), the running operating system (windows, linux, ... to help selecting the proper binaries), may be the hardware architecture or even the operating distribution (for linux based systems). Constraints will filter the search result to include the best, the most accurate version to install. Installation also would configure the installed application and even may run another artifact to help this configuration.

nuts** also handles search for newer versions and update the installed application at request.
Updating a software does not

necessarily delete the older version. Both version can coexist and it is up to the user to decide whether or not to retain both versions. Indeed, one of the key features of **nuts** is the ability to install and hence run multiple versions of the same software in parallel. You would never see an error message telling you can't install that software because a dependency of it is installed with different version. All software and all libraries can coexist peacefully.

Software artifacts are stored in repositories. **nuts** can handle multiple repositories, remote and local ones. Installed software are stored in special local repositories. Supported repositories include maven repositories and github repositories. Actually a fresh installation of **nuts** is configured with maven central repository so that, you already have access to thousands of installable artifacts.

At some point, you may need to uninstall an artifact and that's to undo the artifact installation. Installation will help you choose between uninstalling binaries only and keeping data/config files or remove permanently all of the artifact files. In all ways, uninstalling will not affect other artifacts that use the same dependencies if ever.

Feature rich Toolset

nuts** is intended to be used either by human users or by robots and other applications. It comes with portable,

feature rich toolset, a versatile library and a handy parsable result.

nuts** is mainly a commandline program that helps installing, uninstalling, searching, updating and running artifacts.

To help desktop integration, **nuts** installs by default a set of other companion tools such as **nsh** (a portable bash-compatible implementation), **nadmin** (an administration tool for **nuts** to configure users, authorizations, repositories, ...) and **ndi** (desktop integration) to help creating application shortcuts and scripts;

nsh** brings the bash facilities to all environments (windows included) in a very portable manner. Besides it integrates

well with the installed **nuts** version. Several common commands are ported to **nsh** such as **cat**, **head**, and **ssh**, as well as core features like pipes, redirection and scripts.

nadmin is intended for configuring **nuts**** workspaces, managing repositories and users. It helps also configuring

sub commands and aliases to make **nuts** usage even easier.

ndi**, is the tool for a seamless integration in your operating system. It mainly configures user PATH variable environment and

creates scripts that point to your **nuts** installation on linux or MacOS systems. It creates shortcuts to a pre-configured environment on windows.

Nuts Workspaces

One of the key features of **nuts** is the ability to support multiple isolated workspaces, each managing its own repositories, applications and libraries; each defining its sandbox security constraints. Thus non-root installation is made easy while it remains possible to overlap between workspaces by sharing repositories. Roaming is also supported, so that a workspaces can be

copied/moved across machines.

Application Framework

Nuts** can also be embedded as a library in your application. This enables you to wire classes on the fly by its network

dependency-aware classloading mechanisms. The library allows as well building solid and well integrated applications, mainly console applications. Indeed, **nuts** comes with rich outputs that support automatic formatting to json, xml, table, tree and plain texts. It handles standard File Systems layouts; XDG Base Directory Specification is implemented for linux and MacOS. A compatible one is also implemented in Windows systems. And of course, it helps seamlessly install, update and remove events.

Nuts ? Really ?

In every palace you will find the wizard and the fool, the **maven** and the **nuts** ; There's no exception in the java kingdom! If you do prefer acronyms here is another reason : **nuts** stands for Network Updatable Things Services. It should be able to facilitate things deployment and update over the wire where things resolve here to any piece of software depending (or not) on other piece of software.

Let's start the journey

we start by opening a new terminal (termmm, konsole or whatever you prefer) then download **nuts** using this command : On linux/MacOS system we issue :

```
wget https://github.com/thevpc/vpc-public-
maven/raw/master/net/vpc/app/nuts/nuts/file-version: unsupported file :
/home/vpc/.config/nuts/nuts
Execution Failed with code 2
/nuts-file-version: unsupported file : /home/vpc/.config/nuts/nuts
Execution Failed with code 2
.jar
```

Let's check that java is installed :

```
java --version
```

Now we run **nuts**

```
java -jar nuts-file-version: unsupported file : /home/vpc/.config/nuts/nuts  
Execution Failed with code 2  
.jar -zy
```

We used the flags **-y** to auto-confirm and **-z** to ignore cached binaries (combined here as **-zy**). These flags are not required. We use them here to make installation work in all cases. Installation may last several minutes as it will download all required dependencies, companions and tools.

You should then see this message

```
Welcome to nuts. Yeah, it is working...
```

nuts** is well installed, just restart your terminal.

Now we will install apache tomcat. So in your terminal type:

```
nuts install ntomcat  
nuts ntomcat start --set-port 9090
```

The first two commands will install tomcat helper tool (ntomcat) that will download latest version of tomcat and configure it to 9090 port. The last command will start tomcat. Let's check tomcat status now

```
nuts tomcat status
```

Now we will do the same with derby database. We will install it and run it.

```
nuts install nderby  
nuts nderby start
```

As you can see, simple commands are all you need to download, install, configure and run tomcat or derby or any application that is deployed in the maven repository.

So please visit [nuts website](#) or [github repository](#) for more information. == Repository Structure

```
  \_\_\_ / /_____
 / \_\_ / / / \_\_/
 / \_\_ / / / / (____)
 \_\_ \_\_, / \_\_ /____/    version vfile-version: unsupported file :
/home/vpc/.config/nuts/nuts
Execution Failed with code 2
```

Nuts projects

nuts** repository is composed of several projects that can be organized in 5 categories

Core **nuts** : These projects are the core/base of the **nuts** package manager

Companion Tools : These projects are applications and tools to install with **nuts** itself. Their installation are prompted at first install of Nuts

Toolbox : These projects are applications and tools built on top of **nuts** Application Framework and are of common interest

Lib : These projects are common libraries that can be used to enable some **nuts** features in your application

Other : All other apps that does no fit in the previous categories

1- Core Nuts projects

Core **nuts** projects include nuts-builder, nuts-api, nuts-core. ===== 1.1 nuts-builder

nuts-builder** is a meta project (parent maven pom project) that helps building all the other projects.

1.2 nuts-api

nuts-api is the effective "Nuts" only required dependency. It defines the bootstrap application that is responsible of loading all necessary libraries for its execution. nuts-api starts to load nuts-core which is responsible of implementing all features and interfaces declared by the nuts-api** library. That implementation will handle further download, version dependency etc. Such architecture is considered to force loose coupling with nuts binaries.

nuts-api** is a very thin bootstrapper : its size is about 300k. It can be used as a standalone application or as an embedded library.

1.3 nuts-core

nuts-core is the effective and standard implementation of nuts-api. nuts-core has a faster update pace than nuts-api. It focuses on performance and compliance to the Nuts specification declared by nuts-api interfaces. You are not required to add this dependency to your application if you want to embed Nuts**. The library will be loaded on the wire (if not yet present in the classpath).

nuts-core is designed to have very few dependencies : gson and jansi. gson trivially is used to support json serialization : the main format used in Nuts to support configuration and descriptors. jansi** is used to support terminal coloring and the "Nuts Text Coloring Format" (NTCF), a simple text format that helps creating colorful terminal applications.

2. Companion tools projects

Companion tools include nadmin, nsh and ndi. These three applications are implemented following the " **nuts** Application Framework" and hence are dependent on nuts-api library. They are recommended applications to install with Nuts itself, however they are not mandatory and may be ignored particularly when using nuts-api as library. ===== 2.1 nadmin

nadmin (for **nuts** admin) is an administration tool to the **nuts**** workspaces. It adds support to manage users, credentials, authorizations, workspaces and repositories by providing command line support for such actions.

2.2 nsh

nsh (for **nuts**** shell) is simply a portable POSIX bash compatible implementation. It supports all common builtin commands (ls, cd, rm, ...) and adds support to grep, ssh and scp in a seamless manner. It also supports command line, scripts (including commons constructs with if, do, case, ...) and pipes (|)

2.3 ndi

ndi (for **nuts** desktop integration) is simply a helper tool to support seamless integration of **nuts** commands in your favorite operating system and environment. ndi is responsible of creating script shortcuts to your common commands so that you can invoke them directly from your environment. For instance it creates an "nadmin" script and configures your PATH environment to help calling the nuts admin tool instead of the common way to do so "nuts nadmin". On window system, ndi** will create shortcuts and menus.

3. Toolbox projects

nuts** comes with an array of tools out of the box you can install and play with. Here are these tools

3.1 nded

nded for "**nuts** Descriptor Editor" is a small tool for creating and editing json **nuts**** descriptor file. It is intended to be called by automation tools.

3.2 ncclown

ncclown is an angular web application frontend for **nuts****. It helps navigating, searching and installing artifacts. It is intended to be a web admin tool as well.

3.3 nserver

nserver is a standalone application that runs a small http server that will expose a workspace as a remote repository to other Nuts** installations. This is the simplest way to mirror a workspace and share artifacts between networked nodes.

3.4 nwar

nwar (for **nuts** Web Application Archive) is a web application that exposes nserver** as a war to be deployed on a more mature http server or container.

3.5 ndexer

ndexer (for Indexer) is a lucene powered index for **nuts**. It can be shared across multiple **nuts**** workspaces and processes.

3.6 feenoo

feenoo** is a small search tool. It searches for files, files contents and classes within jars. You can search for files that contain some text or jars that contain some class, or jars of a specific version of java.

The name feenoo comes from the Tunisian dialect and means "where is it?". == 3.7 file-version

file-version** is a small tool that helps detecting files versions. It supports jar, war, ear, dll and exe file versions. It opens a file and looks for its version in its meta-data.

3.8 nmysql

nmysql** is a companion tool to the mysql and mariadb servers. The initial actions supported are backup and restore including push/pull mechanism from/to a couple of databases for synchronization. It supports jdbc and ssh based access to remote mysql/mariadb installation.

3.9 ntomcat

ntomcat** is a companion tool to the tomcat http server. The initial actions supported are start, stop, status, configure (http ports etc..) and deploy. It supports as well installation of several versions of Tomcat and multi domain configuration for deployments.

3.10 nderby

nderby** is a companion tool to the derby database server. The initial actions supported are start, stop, status and configure. It supports as well installation of several versions of Derby.

3.11 nmvn

nmvn** is a companion tool to maven. It supports installations of several versions of it and running them seamlessly.

3.12 worky

worky is a developer centered tool. The 'y' in worky refers to 'my' in the "Tunisian Dialect" and hence means "my work". Worky is the tool we - maven users - need to check if the version of project we are working on is yet to be deployed to nexus or not. So basically it checks if the version is the same, and downloads the server's version and then compares binaries to local project's to check if we have missed to update the version in our pom.xml. I know I'm not the only one having pain with jar deployments to nexus. Worky** does other things as well to help me on daily basis.

4. Library Projects

Library projects are several libraries that add **nuts** support in a particular environment or domain.

==== 4.1 nuts-tomcat-classloader This is a must-have feature in your web application if deployed on Tomcat. It solves the following problem : a simple war application is surprisingly fat with too many jars (hundreds of Megas) you need to upload each time you change a single file or class in your web project. Basically all the jars included in the lib folder of the war are to be uploaded each time to the remote Tomcat server. The common solution is to use "provided" scope in maven and put your jars in Tomcat lib or ext folders. This is a bad approach if you are using a single Tomcat process for multiple applications. nuts-tomcat-classloader simply uses Nuts to download libraries when the application is deployed based on the pom.xml you provide and include them in the current web application class loader. Hence, the war becomes lighter than ever. Nuts cache mechanisms optimizes bandwidth and makes this more convenient by sharing the same jar files between applications depending on the same versions. All you have to do is to add this library to your application and configure your pom.xml accordingly.

==== 4.2 nuts-servlet Basically this is the simplest way to include nserver into your web application.

==== 4.3 nuts-lib-template This library provides helper methods to manipulate maven pom.xml and generate simple Java files while supporting **nuts** concepts. It is used in other tools that are meant to generate maven projects.

== 5. Other Projects Other projects you may encounter in the repository are WIP projects that may be continued or discontinued. This includes : nutsc (a native c bootstrapper) and nuts-installer (a Nuts

installer tool)

6. Honorable mentions

Although not included in this Git repository some other tools are based on `nuts` and hence are installable using `nuts install the-app` command. Those tools are published in other repositories.

6.1 netbeans-launcher : this tool supports installation and launch of multiple netbeans instances in parallel. See [Netbeans Launcher GitHub Repository](#)

6.2 upa-box : this tool supports creation of UPA aware projects. UPA is a non structured ORM for the Java Language. See [Netbeans Launcher GitHub Repository](#)

6.3 vr-box : this tool supports creation of VR aware projects. VR is a web portal framework. See [Netbeans Launcher GitHub Repository](#)

Getting Started

Command Line Arguments

```
  /\ \ \ \ _ _/_ /_ _  
  / \ \ / / / / _/_/  
  / / \ / / / / _(_ )  
 \_ \ \ \_, / \_/_/    version vfile-version: unsupported file :  
 /home/vpc/.config/nuts/nuts  
 Execution Failed with code 2
```

Nuts Commandline

Nuts supports a specific format for command line arguments. This format is the format supported in **nuts**** Application Framework (NAF) and as such all NAF applications support the same command line arguments format.

Arguments in **nuts** can be options or non options. Options always start with dash (-). === Short vs Long Options Options can be long options (starts with double dash) or short options (start with a single dash). Many arguments support both forms. For instance "-w" and "--workspace" are the supported forms to define the workspace location in the nuts command.

Option Values

Options can also support a value of type string or boolean. The value can be suffixed to the option while separated with '=' sign or immediately after the option. As an example "-w=/myfolder/myworkspace" and "--workspace /myfolder/myworkspace" are equivalent.

Boolean Options

Particularly, when the value is a boolean, the value do not need to be defined. As a result "--skip-companions" and "--skip-companions=true" are equivalent. However "--skip-companions true" is not (because the option is of type boolean) and "true" will be parsed as a NonOption.

To define a "false" value to the boolean option we can either suffix with "=false" or prefix with "!" or "~" sign. Hence, "--skip-companions=false", "--!skip-companions" and "--~skip-companions" are all equivalent.

Combo Simple Options

Simple options can be grouped in a single word. "-ls" is equivalent to "-l -s". So one should be careful. One exception though. For portability reasons, "-version" is considered a single short option.

Ignoring Options, Comments

Options starting with "-//:" and "--//:" are simply ignored by the command line parser.

Nuts Option Types

Options in **nuts** are regrouped in multiple categories. An option can belong to multiple categories though.

Create Options : such options are only relevant when creating a new workspace. They define the configuration of the workspace to create. They will be ignored when the workspace already exists. Examples include

- --skip-companions
- --archetype
- --store-strategy
- --standalone

Open Options : such options are relevant when creating a new workspace or when opening an existing workspace. They define the way commands are executed. Examples include

- --workspace
- --bot
- --reset

Exported Options : are passed to sub-nuts-processes that will be created by nuts. For instance when nuts will call the nsh command it will spawn a new process. In such case, these options are passed to the sub-process as environment variable.

- --workspace
- --bot
- --no-color

Application Options : are options that are by default supported by Applications using NAF (Nuts Application Framework) (as well as Nuts it self).

- --help
- --version

all **nuts** options are described in the command help. Just type :

```
nuts --help
```

Frequently Asked Questions

```
  \_\_\_ / / / / / / / /  
  / \_\_\_ / / / / / / / /  
  / \_\_\_ / / / / / / ( )  
  \_\_\_ / \_\_\_ / \_\_\_ /   version vfile-version: unsupported file :  
  /home/vpc/.config/nuts/nuts  
Execution Failed with code 2
```

Why do we need a package manager for Java. Isn't Maven enough?

please read [Nuts Introduction, Why and What for](#) === What does Nuts mean and why ?

nuts** stands for "Network Updatable Things Services". It helps managing things (artifacts of any type, not only java).

The Name also helps depicting another idea : **nuts** is a good companion and complement to Maven tool. The word maven (MAY-vin), from Yiddish, means a super-enthusiastic expert/fan/connoisseur/Wizard. And where wizards are, fools and **nuts** must be. **nuts** is the foolish tool to support the deployment and not the build. Hence the name.

Does nuts support only jar packaging

Not only. **nuts** supports all packagings supported by maven. This includes pom , jar , maven-plugin , ejb , war , ear , rar. However **nuts** is also intended to support any "thing" including "exe" , "dll" , "so" , "zip" files, etc. === **nuts**** differs from maven as it defines other properties to the artifact descriptor (aka pom in maven) : os (operating system), arch (hardware architecture), osdist (relevant for linux for instance : opensuse, ubuntu) and platform (relevant to vm platforms like java vm, dotnet clr, etc). Such properties are queried to download the most appropriate binaries for the the current characteristics.

Can I contribute to the project

I hoped you would ask this question. Sure. You can drop me an email to add you as contributor or fork the repository and ping a pull request. You can also open a new issue for feature implementation to invite any other contributor to implement that feature (or even implement it your self).

Where can I find Documentation about the Project

The doc folder is intended to include documentation. The wiki also should help.

How can I make my application "Nuts aware"

If by **nuts** aware you mean that you would download your application and run it using **nuts**, then you just need to create the application using maven and deploy your application to the public maven central. Nothing really special is to be done from your side. You do not have to use plugins like 'maven-assembly-plugin' and 'maven-shade-plugin' to include your dependencies. Or, you can also use NAF (**nuts** Application Framework) make your application full featured "Nuts aware" application.

Why should I consider implementing my terminal application using Nuts Application Framework (NAF)

First of all, NAF is a simple 300k jar so for what it provided to you, you would be surprised. Indeed, implementing your application using NAF will provide you a clean way to :

seamless integration with **nuts** and all other NAF applications

support standard file system layout (XDG) where config files and log files are not necessarily in the same folder see [Nuts File System](../advanced/filesystem.md) for more details.

support application life cycle events (onInstall, onUninstall, onUpgrade),

standard support of command line arguments

dynamic dependency aware class loading

terminal coloring and components (progress bar, etc...)

json,xml,table,tree and plain format support out of the box

pipe manipulation when calling sub processes

advanced io features (persistence Locks, monitored downloads, compression, file hashing....)

standard ways to support and use installed platforms (installed JRE, JDK, ...)

and more...

Can I use NAF for non terminal applications, Swing or JavaFX perhaps

Sure, you will be able to benefit of all the items in the preceding question but terminal coloring wont be relevant of course. Check netbeans-launcher in github. It's a good example of how interesting is to use NAF in non terminal applications.

What is the License used in Nuts

nuts** is published under GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later.

Installation

```

  /\_/\_/_/ /_/
 / \_/_/_/_/_/ _/ _/_
 / \_/_/_/_/_(_ _ )
 \_/\_/_/_/_/_/   version vfile-version: unsupported file :
 /home/vpc/.config/nuts/nuts
 Execution Failed with code 2

```

System Requirements

Here are all **nuts** requirements : * Java : **nuts** requires a valid Java Runtime Environment (JRE) or Java Development Kit (JDK) version 8 or above to execute.

- System Memory: **nuts** memory footprint is very little and has no minimum RAM requirements.
- Disk: 2.5Mo on the disk are required for the **nuts** installation itself. In addition to that, additional disk space will be used for your local Nuts workspace. The size of your local workspace will vary depending on usage but expect at least 500MB.
- Operating System: **nuts** is able to run on any java enabled Operating System including all recent versions of Windows, Linux and MacOS. To check if you have a valid java installation type

```
java -version
```

The result would be equivalent to the following. Just be sure the version is 1.8 or over. In this example, the java version is 1.8.0_211

```

$> java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)

```

Installation

Linux

```
NDVER=file-version: unsupported file : /home/vpc/.config/nuts/nuts
Execution Failed with code 2
&& curl -OL https://github.com/thevpc/vpc-public-maven/raw/master\
/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -zy
```

Linux Systems installation is based on bash shell. First launch will configure "~/.bashrc" so that nuts and other companion tool commands will be available in any future terminal instances. Using **nuts** on unix-like system should be seamless. A simple bash terminal (Gnome Terminal, KDE Konsole,...) is already a nuts-aware terminal. All Linux versions and distributions should work with or without XWindow (or equivalent). Graphical system is required only if you plan to run a gui application using nuts. All tests were performed on OpenSuse Tumbleweed.

TIP: Any bash terminal application is a nuts-aware terminal.

MacOS

```
NDVER=file-version: unsupported file : /home/vpc/.config/nuts/nuts
Execution Failed with code 2
&& curl -OL https://github.com/thevpc/vpc-public-maven/raw/master\
/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -y
```

MacOS Systems installation is based on bash shell. First launch will configure "~/.bashrc" so that nuts and other companion tool commands will be available in any future terminal instances. Using **nuts** on MacOS system should be seamless. A simple bash terminal (MacOs Terminal App) is already a nuts-aware terminal.

TIP: Any bash terminal application is a nuts-aware terminal.

Windows

```
download [nuts-file-version: unsupported file : /home/vpc/.config/nuts/nuts[Execution Failed with
code 2 .jar](https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/
0.file-version: unsupported file : /home/vpc/.config/nuts/nuts Execution Failed with code 2 /nuts-
file-version: unsupported file : /home/vpc/.config/nuts/nuts Execution Failed with code 2 .jar)
```

```
java -jar -y nuts-file-version: unsupported file : /home/vpc/.config/nuts/nuts
Execution Failed with code 2
.jar
```

On Windows systems, first launch will create a new **nuts** Menu (under Programs) and a couple of Desktop shortcuts to launch a configured command terminal.

* nuts-cmd-file-version: unsupported file : /home/vpc/.config/nuts/nuts Execution Failed with code 2

: this shortcut will open a configured command terminal. **nuts** command will be available as well as several nuts companion tools installed by ndi by default

- nuts-cmd : this shortcut will point to the last installed nuts version, here file-version: unsupported file : /home/vpc/.config/nuts/nuts Execution Failed with code 2

Any of these shortcuts will launch a nuts-aware terminal.

Supported Windows systems include Window 7 and later.

TIP: Any of the created shortcuts for windows is a nuts-aware terminal.

*NIX wget

```
NDVER=file-version: unsupported file : /home/vpc/.config/nuts/nuts
Execution Failed with code 2
&& rm -f nuts-$NDVER.jar && wget https://github.com/thevpc/\
vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar && \
java -jar nuts-$NDVER.jar -y
```

TIP: Any bash terminal application is a nuts-aware terminal.

*NIX curl

```
NDVER=file-version: unsupported file : /home/vpc/.config/nuts/nuts
Execution Failed with code 2
&& curl -OL https://github.com/thevpc/vpc-public-maven/raw/master\
/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -y
```

TIP: Any bash terminal application is a nuts-aware terminal.

Any Java enabled OS

```
NDVER=file-version: unsupported file : /home/vpc/.config/nuts/nuts
Execution Failed with code 2
&& curl -OL https://github.com/thevpc/vpc-public-maven/raw/master\
/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -y
```

TIP: Any bash terminal application is a nuts-aware terminal.

You should then see some log like the following :

As you can see, installation upon first launch, will also trigger installation of other optional programs called "companion tools". Actually they are recommended helpful tools : + ndi which stands for Nuts Desktop Integration that helps configuring the desktop to better interact with **nuts** by for instance creating shortcuts. + nsh which stands for Nuts Shell , a bash compatible shell implementation program that will run equally on linux an windows systems. + nadmin an administration tool for **nuts**

IMPORTANT: After installation, you need to restart the terminal application for the configuration to take effect.

Test Installation

To test installation the simplest way is to open a nuts-aware terminal and type :

```
nuts --version
```

It should show a result in the format : nuts-api-version/nuts-impl-version

```
file-version: unsupported file : /home/vpc/.config/nuts/nuts
Execution Failed with code 2
/nuts nversion ..//nuts-code
```

Run a command

To run a command using nuts just type

```
nuts <command>
```

Several commands are available, and you still be able to run any java and non java application. More info is available in the [nuts](#) official [wiki](#). == Running Nuts

```
/\ \ \ \_ _/_ /_____
/\ \ / / / / / _/_/
/\ / / / / / /(_ _ )
\_ \ \_\_,/\_\_/_/    version vfile-version: unsupported file :
/home/vpc/.config/nuts/nuts
Execution Failed with code 2
```

Running a deployed artifact

You can run any jar using **nuts** as far as the jar is accessible from one of the supported repositories. By default, **nuts** supports: + maven central + local maven folder (~/.m2)

You can configure other repositories or even implement your own if you need to.

The jar will be parsed to check form maven descriptor so that dependencies will be resolved and downloaded on the fly. Then, all executable classes (public with static void main method) are enumerated. You can actually run any of them when prompted. Any jar built using maven should be well described and can be run using its artifact long id.

Artifact Long Ids

nuts** long ids are a string representation of a unique identifier of the artifact. It has the following form :

```
groupId:artifactId#version
```

for instance, to install **netbeans-launcher** (which is a simple UI helping launch of multiple

instances of netbeans), you can issue

```
nuts net.vpc.app:netbeans-launcher#1.2.2
```

You do agree that this can be of some cumbersome to type. So you can simplify it to :

```
nuts netbeans-launcher
```

In this form, **nuts** will auto-detect both the **groupId** and the **version**. The group id is detected if it is already imported (we will see later import a groupId). By default, there is a couple of groupIds that are automatically imported :

```
+ 'net.vpc.app' (contains various applications of the author) +  
'net.vpc.nuts.toolbox' (contains various companion tools of ** 'nuts' **, such  
as 'nsh' , 'nadmin' , 'ndi' , ...)  
And it turns out, hopefully, that netbeans-launcher belongs to an imported  
groupId, so we can omit it.  
Besides, if no version is provided, ** 'nuts' ** will also auto-detect the best  
version to execute. If the application is already installed, the version you  
choose to install will be resolved. If you have not installed any, the most  
recent version, obviously, will be detected for you.
```

Artifact Installation

Any java application can run using **nuts** but it has to be installed first. If you try to run the application before installing it, you will be prompted to confirm installation. To install our favorite application here we could have issued :

```
nuts install netbeans-launcher
```

But as we have tried to run the application first, it has been installed for us (after confirmation).

Multiple Artifact version Installation

One of the key features of **nuts** is the ability to install multiple versions of the same application. We can for instance type :

```
nuts install netbeans-launcher#1.2.2
# then
nuts install netbeans-launcher#1.2.0
```

Now we have two versions installed, the last one always is considered default one. you can run either, using it's version

```
nuts netbeans-launcher#1.2.2 &
# or
nuts netbeans-launcher#1.2.0 &
```

Actually, when you have many versions installed for the same artifact and you try to run it without specifying the version, the last one installed will be considered. To be more precise, an artifact has a default version when it is installed. This default version is considered when no explicit version is typed. In our example, when we type

```
nuts netbeans-launcher &
```

the 1.2.0 version will be invoked because the artifact is already installed and the default version points to the last one installed. So if you want to switch back to version 1.2.2 you just have to re-install it. Don't worry, no file will be downloaded again, nuts will detect that the version is not marked as default and will switch it to.

Searching artifacts

Now let's take a look at installed artifacts. We will type :

```
nuts search --installed
```

This will list all installed artifacts. We can get a better listing using long format :

```
nuts search --installed -l
```

you will see something like

```
I-X 2019-08-21 04:54:22.951 anonymous vpc-public-maven net.vpc.app:netbeans-launcher#1.2.0
i-X 2019-08-21 04:54:05.196 anonymous vpc-public-maven net.vpc.app:netbeans-launcher#1.2.2
```

The first column here is the artifact status that helps getting zipped information of the artifact. The 'I' stands for 'installed and default' whereas, 'i' is simply 'installed'. The 'X' stands for 'executable application', where 'x' is simply 'executable'. Roughly said, executable applications are executables aware of (or depends on) nuts, as they provide a special API that helps nuts to get more information and more features for the application. As an example, executable applications have special OnInstall and OnUninstall hooks called by nuts. The second and the third columns are date and time of installation. The fourth column points to the installation user. When Secure mode has not been enabled (which is the default), you are running nuts as 'anonymous'. The fifth column shows the repository from which the package was installed. And the last column depicts the artifact long id.

Running local jar file with its dependencies

Let's suppose that my-app.jar is a maven created jar (contains META-INF/maven files) with a number of dependencies. **nuts** is able to download on the fly needed dependencies, detect the Main class (no need for MANIFEST.MF) and run the application. If a Main-Class Attribute was detected in a valid MANIFEST.MF, it will be considered. If more than one class is detected with a main method, **nuts** will ask for the current class to run. When you run a local file, **nuts** will behave as if the app is installed (in the given path, no need to invoke install command). Local files are detected if they are denoted by a valid path (containing '/' or '\' depending on the underlying operating system). Dependencies will be downloaded as well (and cached in the workspace)

```
nuts ./my-app.jar some-argument-of-my-app
```

If you need to pass JVM arguments you have to prefix them with "--exec". So if you want to fix maximum heap size use

```
nuts --exe -Xms1G -Xmx2G ./my-app.jar argument-1 argument-2
```

Application Framework

Nuts Applications

```
  /\ \ \ \ _ _/_ /_ _  
 / \ \ / / / / _/_/  
 / / \ / / / / _(_ )  
 \_\ \ \_,/_\ /_ _/_/  version vfile-version: unsupported file :  
 /home/vpc/.config/nuts/nuts  
Execution Failed with code 2
```

Your first Nuts Application

```
  /\ \ \ \ _ _/_ /_ _  
 / \ \ / / / / _/_/  
 / / \ / / / / _(_ )  
 \_\ \ \_,_/\_/_/_/   version vfile-version: unsupported file :  
 /home/vpc/.config/nuts/nuts  
Execution Failed with code 2
```

[API Reference](#)

Application

☕ NutsApplication

```
public abstract net.vpc.app.nuts.NutsApplication
```

Nuts Application is the Top Level class to be handled by nuts as rich console application. By default NutsApplication classes :

- have a nutsApplication=true in their descriptor file
- support inheritance of all workspace options (from caller nuts process)
- enables auto-complete mode to help forecasting the next token in the command line
- enables install mode to be executed when the jar is installed in nuts repos
- enables uninstall mode to be executed when the jar is uninstalled from nuts repos
- enables update mode to be executed when a new version of the same jar has been installed
- have many default options enabled (such as --help, --version, --json,
 - table, etc.) and thus support natively multi output channels

Typically a Nuts Application follows this code pattern :

```

public class MyApplication extends NutsApplication{
    public static void main(String[] args) {
        // just create an instance and call runAndExit in the main method
        new MyApplication().runAndExit(args);
    }
    // do the main staff in launch method
    public void run(NutsApplicationContext appContext) {
        boolean myBooleanOption=false;
        NutsCommandLine cmdLine=appContext.getCommandLine()
        boolean boolOption=false;
        String stringOption=null;
        Argument a;
        while(cmdLine.hasNext()){
            if(appContext.configureFirst(cmdLine)){
                //do nothing
            }else {
                a=cmdLine.peek();
                switch(a.getStringKey()){
                    case "-o": case "--option":{
                        boolOption=cmdLine.nextBoolean().getBooleanValue();
                        break;
                    }
                    case "-n": case "--name":{
                        stringOption=cmdLine.nextString().getStringValue();
                        break;
                    }
                    default:{
                        cmdLine.unexpectedArgument();
                    }
                }
            }
        }
        // test if application is running in exec mode
        // (and not in autoComplete mode)
        if(cmdLine.isExecMode()){
            //do the good staff here
        }
    }
}

```

another example of using this class is :

```

public class HLMain extends NutsApplication {
    public static void main(String[] args) {

```

```

// just create an instance and call runAndExit in the main method
new HLMain().runAndExit(args);
}

@Override
public void run(NutsApplicationContext applicationContext) {
    applicationContext.processCommandLine(new NutsCommandLineProcessor())
{
    HLCWithOptions hl = new HL().withOptions();
    boolean noMoreOptions=false;
    @Override
    public boolean processOption(NutsArgument argument,
NutsCommandLine cmdLine) {
        if(!noMoreOptions){
            return false;
        }
        switch (argument.getStringKey()) {
            case "--clean": {
                hl.clean(cmdLine.nextBoolean().getBooleanValue());
                return true;
            }
            case "-i":
            case "--incremental":{

                hl.setIncremental(cmdLine.nextBoolean().getBooleanValue());
                return true;
            }
            case "-r":
            case "--root":{

                hl.setProjectRoot(cmdLine.nextString().getStringValue());
                return true;
            }
        }
        return false;
    }

    @Override
    public boolean processNonOption(NutsArgument argument,
NutsCommandLine cmdLine) {
        String s = argument.getString();
        if(isURL(s)){
            hl.includeFileURL(s);
        }else{
            hl.includeFile(s);
        }
    }
}

```

```
        noMoreOptions=true;
        return true;
    }

    private boolean isURL(String s) {
        return
            s.startsWith("file:")
            ||s.startsWith("http:")
            ||s.startsWith("https:")
            ;
    }

    @Override
    public void exec() {
        hl.compile();
    }
};

}
```

⌚⚙️ Static Methods

⌚⚙️ main(appType, args)

creates an instance of `appType` and calls runAndExit. This method is intended be called in main methods of NutsApplication classes.

```
void main(Class<T> appType, String[] args)
```

application type main arguments

⚙️ Instance Methods

⚙️ createApplicationContext(ws, args, startTimeMillis)

create application context or return null for default

```
NutsApplicationContext createApplicationContext(NutsWorkspace ws, String[] args,
                                             long startTimeMillis)
```

workspace arguments start time

⚙️ `onInstallApplication(applicationContext)`

this method should be overridden to perform specific business when application is installed

```
void onInstallApplication(NutsApplicationContext applicationContext)
```

context

⚙️ `onUninstallApplication(applicationContext)`

this method should be overridden to perform specific business when application is uninstalled

```
void onUninstallApplication(NutsApplicationContext applicationContext)
```

context

⚙️ `onUpdateApplication(applicationContext)`

this method should be overridden to perform specific business when application is updated

```
void onUpdateApplication(NutsApplicationContext applicationContext)
```

context

⚙️ `run(applicationContext)`

run application within the given context

```
void run(NutsApplicationContext applicationContext)
```

app context

⚙️ `run(args)`

run the application with the given arguments. If the first arguments is in the form of --nuts-exec

-mode=... the argument will be removed and the corresponding mode is activated.

```
void run(String[] args)
```

application arguments. should not be null or contain nulls

⚙️ `run(session, args)`

run the application with the given arguments against the given workspace If the first arguments is in the form of --nuts-exec-mode=... the argument will be removed and the corresponding mode is activated.

```
void run(NutsSession session, String[] args)
```

session (can be null) application arguments. should not be null or contain nulls

⚙️ `runAndExit(args)`

run the application and EXIT process

```
void runAndExit(String[] args)
```

arguments

⚙️ `toString()`

```
String toString()
```

🔩 `NutsApplicationContext`

```
public net.vpc.app.nuts.NutsApplicationContext
```

Application context that store all relevant information about application execution mode, workspace, etc.

Instance Fields

AUTO_COMPLETE_CANDIDATE_PREFIX

```
String AUTO_COMPLETE_CANDIDATE_PREFIX = "@@Candidate@@: "
```

Instance Properties

appClass

application class reference

```
[read-only] Class appClass  
Class getAppClass()
```

appId

application nuts id

```
[read-only] NutsId appId  
NutsId getAppId()
```

appPreviousVersion

previous version (applicable in update mode)

```
[read-only] NutsVersion appPreviousVersion  
NutsVersion getAppPreviousVersion()
```

appVersion

application version

```
[read-only] NutsVersion appVersion  
NutsVersion getAppVersion()
```

appsFolder

path to the apps folder of this application

```
[read-only] Path appsFolder  
Path getAppsFolder()
```

arguments

application arguments

```
[read-only] String[] arguments  
String[]  getArguments()
```

autoComplete

Auto complete instance associated with the ` NutsApplicationMode#AUTO_COMPLETE` mode

```
[read-only] NutsCommandAutoComplete autoComplete  
NutsCommandAutoComplete getAutoComplete()
```

cacheFolder

path to the cache files folder of this application

```
[read-only] Path cacheFolder  
Path getCacheFolder()
```

commandLine

a new instance of command line arguments to process filled with application's arguments.

```
[read-only] NutsCommandLine commandLine  
NutsCommandLine getCommandLine()
```

configFolder

path to the configuration folder of this application

```
[read-only] Path configFolder  
Path getConfigFolder()
```

execMode

return true if `getAutoComplete()==null`

```
[read-only] boolean execMode  
boolean isExecMode()
```

libFolder

path to the libraries files (non applications) folder of this application

```
[read-only] Path libFolder  
Path getLibFolder()
```

logFolder

path to the log folder of this application

```
[read-only] Path logFolder  
Path getLogFolder()
```

mode

application execution mode

```
[read-only] NutsApplicationMode mode  
NutsApplicationMode getMode()
```

modeArguments

application execution mode extra arguments

```
[read-only] String[] modeArguments  
String[] getModeArguments()
```

runFolder

path to the temporary run files (non essential sockets etc...) folder of this application

```
[read-only] Path runFolder  
Path getRunFolder()
```

session

update session

```
[read-write] NutsApplicationContext session  
NutsSession getSession()  
NutsApplicationContext setSession(session)
```

sharedAppsFolder

```
[read-only] Path sharedAppsFolder  
Path getSharedAppsFolder()
```

sharedConfigFolder

```
[read-only] Path sharedConfigFolder  
Path getSharedConfigFolder()
```

sharedLibFolder

[read-only] Path sharedLibFolder
Path **getSharedLibFolder()**

📄📁 sharedLogFolder

[read-only] Path sharedLogFolder
Path **getSharedLogFolder()**

📄📁 sharedRunFolder

[read-only] Path sharedRunFolder
Path **getSharedRunFolder()**

📄📁 sharedTempFolder

[read-only] Path sharedTempFolder
Path **getSharedTempFolder()**

📄📁 sharedVarFolder

[read-only] Path sharedVarFolder
Path **getSharedVarFolder()**

📄⏱ startTimeMillis

application start time in milli-seconds

[read-only] long startTimeMillis
long **getStartTimeMillis()**

📄📁 tempFolder

path to the temporary files folder of this application

```
[read-only] Path tempFolder  
Path getTempFolder()
```

📄 varFolder

path to the variable files (aka /var in POSIX systems) folder of this application

```
[read-only] Path varFolder  
Path getVarFolder()
```

📄 workspace

current workspace

```
[read-only] NutsWorkspace workspace  
NutsWorkspace getWorkspace()
```

⚙️ Instance Methods

⚙️ configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsApplicationContext configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ configureLast(commandLine)

calls configureFirst and ensure this is the last test

```
boolean configureLast(NutsCommandLine commandLine)
```

arguments to configure with

⚙️ `getFolder(location)`

application store folder path for the given `location`

Path `getFolder(NutsStoreLocation location)`

location type

⚙️ `getSharedFolder(location)`

Path `getSharedFolder(NutsStoreLocation location)`

null

⚙️ `printHelp()`

print application help to the default out (``getSession().out()``) print stream.

`void printHelp()`

⚙️ `processCommandLine(commandLineProcessor)`

create new NutsCommandLine and consume it with the given processor. This method is equivalent to the following code

```

NutsCommandLine cmdLine=getCommandLine();
NutsArgument a;
while (cmdLine.hasNext()) {
    if (!this.configureFirst(cmdLine)) {
        a = cmdLine.peek();
        if(a.isOption()){
            if(!commandLineProcessor.processOption(a,cmdLine)){
                cmdLine.unexpectedArgument();
            }
        }else{
            if(!commandLineProcessor.processNonOption(a,cmdLine)){
                cmdLine.unexpectedArgument();
            }
        }
    }
}
// test if application is running in exec mode
// (and not in autoComplete mode)
if (cmdLine.isExecMode()) {
    //do the good stuff here
    commandLineProcessor.exec();
}

```

This as an example of its usage

```

applicationContext.processCommandLine(new NutsCommandLineProcessor() {
    HLCWithOptions hl = new HL().withOptions();
    boolean noMoreOptions=false;
    @Override
    public boolean processOption(NutsArgument argument, NutsCommandLine
cmdLine) {
        if(!noMoreOptions){
            return false;
        }
        switch (argument.getStringKey()) {
            case "--clean": {
                hl.clean(cmdLine.nextBoolean().getBooleanValue());
                return true;
            }
            case "-i":
            case "--incremental":{

hl.setIncremental(cmdLine.nextBoolean().getBooleanValue());
                return true;
            }
        }
    }
}

```

```

        }
        case "-r":
        case "--root":{
            hl.setProjectRoot(cmdLine.nextString().getStringValue());
            return true;
        }
    }
    return false;
}

@Override
public boolean processNonOption(NutsArgument argument,
NutsCommandLine cmdLine) {
    String s = argument.getString();
    if(isURL(s)){
        hl.includeFileURL(s);
    }else{
        hl.includeFile(s);
    }
    noMoreOptions=true;
    return true;
}

private boolean isURL(String s) {
    return
        s.startsWith("file:")
        ||s.startsWith("http:")
        ||s.startsWith("https:")
        ;
}
}

@Override
public void exec() {
    hl.compile();
}
});

```

```
void processCommandLine(NutsCommandLineProcessor commandLineProcessor)
```

commandLineProcessor

☕ NutsApplicationLifeCycle

```
public net.vpc.app.nuts.NutsApplicationLifeCycle
```

Application Life Cycle interface define methods to be overridden to perform specific business for each of the predefined application execution modes ` NutsApplicationMode` . ===== ⚙ Instance Methods

⚙ `createApplicationContext(ws, args, startTimeMillis)`

this method should be implemented to create specific ApplicationContext implementation or return null to use default one.

```
NutsApplicationContext createApplicationContext(NutsWorkspace ws, String[] args,  
long startTimeMillis)
```

workspace application arguments start time in milliseconds

⚙ `onInstallApplication(applicationContext)`

this method should be implemented to perform specific business when application is installed.

```
void onInstallApplication(NutsApplicationContext applicationContext)
```

context

⚙ `onRunApplication(applicationContext)`

this method should be implemented to perform specific business when application is running (default mode)

```
void onRunApplication(NutsApplicationContext applicationContext)
```

context

⚙️ onUninstallApplication(applicationContext)

this method should be implemented to perform specific business when application is un-installed.

```
void onUninstallApplication(NutsApplicationContext applicationContext)
```

context

⚙️ onUpdateApplication(applicationContext)

this method should be implemented to perform specific business when application is updated.

```
void onUpdateApplication(NutsApplicationContext applicationContext)
```

context

☕ NutsApplications

```
public final net.vpc.app.nuts.NutsApplications
```

Helper class for Nuts Applications

CONSTANT FIELDS

Constant Fields

Constant Fields

Constant Fields

```
private static final ThreadLocal<Map<String, Object>> sharedMap = new  
ThreadLocal<>()
```

STATIC PROPERTIES

sharedMap

a thread local map used to share information between workspace and embedded applications.

```
[read-only] public static Map<String, Object> sharedMap  
public static Map<String, Object> getSharedMap()
```

⌚⚙️ Static Methods

⌚⚙️ processThrowable(ex, args, out)

process throwables and return exit code

```
int processThrowable(Throwable ex, String[] args, PrintStream out)
```

exception application arguments to check from if a '--verbose' or '--debug' option is armed out stream

⌚⚙️ runApplication(args, session, appClass, lifeCycle)

run application with given life cycle.

```
void runApplication(String[] args, NutsSession session, Class appClass,  
NutsApplicationLifeCycle lifeCycle)
```

application arguments session application class application life cycle

⌚ Constructors

⌚ NutsApplications()

private constructor

```
NutsApplications()
```

Base

Nuts

```
public final net.vpc.app.nuts.Nuts
```

Nuts Top Class. Nuts is a Package manager for Java Applications and this class is it's main class for creating and opening nuts workspaces.

Static Properties

platformOsFamily

default OS family, resolvable before booting nuts workspace

```
[read-only] public static NutsOsFamily platformOsFamily
public static NutsOsFamily getPlatformOsFamily()
```

version

current Nuts version

```
[read-only] public static String version
public static String version
public static String getVersion()
```

Static Methods

getPlatformHomeFolder(storeLocationLayout, folderType, homeLocations, global, workspaceName)

resolves nuts home folder.Home folder is the root for nuts folders.It depends on folder type and store layout. For instance log folder depends on the underlying operating system (linux,windows,...). Specifications: XDG Base Directory Specification (<https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html>)

```
String getPlatformHomeFolder(NutsOsFamily storeLocationLayout, NutsStoreLocation folderType, Map<String, String> homeLocations, boolean global, String workspaceName)
```

location layout to resolve home for folder type to resolve home for workspace home locations global workspace workspace name or id (discriminator)

main(args)

main method. This Main will call ` Nuts#runWorkspace(java.lang.String...)` then ` System#exit(int)` at completion

```
void main(String[] args)
```

main arguments

openInheritedWorkspace(args)

opens a workspace using "nuts.boot.args" and "nut.args" system properties. "nuts.boot.args" is to be passed by nuts parent process. "nuts.args" is an optional property that can be 'exec' method. This method is to be called by child processes of nuts in order to inherit workspace configuration.

```
NutsWorkspace openInheritedWorkspace(String args)
```

arguments

openWorkspace()

open default workspace (no boot options)

```
NutsWorkspace openWorkspace()
```

openWorkspace(args)

open a workspace. Nuts Boot arguments are passed in `` args

NutsWorkspace openWorkspace(String args)

nuts boot arguments

☕⚙️ openWorkspace(options)
open a workspace using the given options

NutsWorkspace openWorkspace(NutsWorkspaceOptions options)

boot options

☕⚙️ parseNutsArguments(bootArguments)
Create a ``` NutsWorkspaceOptions``` instance from string array of valid
nuts options

NutsWorkspaceOptions parseNutsArguments(String[] bootArguments)

input arguments to parse

☕⚙️ runWorkspace(args)
open then run Nuts application with the provided arguments. This Main
will
NEVER
call ``` System#exit(int)```.

void runWorkspace(String args)

boot arguments

⚒ Constructors
🌐 Nuts()
private constructor

Nuts()

🍵 NutsArtifactCall

public net.vpc.app.nuts.NutsArtifactCall

```
artifact call descriptor used to define executor and installer call definitions.  
### □ Instance Properties  
#### 📄 □ arguments  
execution arguments
```

[read-only[String[] arguments String[] getArguments()

```
#### 📄 □ id  
artifact id
```

[read-only[NutsId id NutsId getId()

```
#### 📄 □ properties  
execution properties
```

[read-only[Map<String,String> properties Map<String,String> getProperties()

```
## ☕ NutsContentEvent
```

public net.vpc.app.nuts.NutsContentEvent

```
Event for `NutsRepositoryListener` methods.  
### □ Instance Properties  
#### 📄 □ path  
artifact path
```

[read-only[Path path Path getPath()

```
#### 📄 □ repository  
current repository
```

[read-only[NutsRepository repository NutsRepository getRepository()

📄 session
current session

[read-only] NutsSession session NutsSession getSession()

📄 workspace
current workspace

[read-only] NutsWorkspace workspace NutsWorkspace getWorkspace()

☕ NutsDefinition

public net.vpc.app.nuts.NutsDefinition

Definition is an ****immutable**** object that contains all information about a artifact identified by it's Id.

Instance Properties

📄 apiId

return target api id (included in dependency) for the current id.

This is relevant for runtime, extension and companion ids.

For other regular ids, this returns null.

[read-only] NutsId apild NutsId getApild()

📄 content

return artifact content file info (including path).

this is an ****optional**** property. It must be requested (see `'''
NutsSearchCommand#setContent(boolean)'''` to be available.

[read-only] NutsContent content NutsContent getContent()

📄 dependencies

return all or some of the transitive dependencies of the current Nuts as List result of the search command

this is an ****optional**** property.

It must be requested (see `''' NutsSearchCommand#setDependencies(boolean)'''` to be available.

[read-only[NutsDependency[[dependencies NutsDependency[[getDependencies()

```
####  dependencyNodes  
return all of some of the transitive dependencies of the current Nuts as Tree  
result of the search command  
this is an **optional** property.  
It must be requested (see `NutsSearchCommand#setDependenciesTree(boolean)`)  
to be available.
```

[read-only[NutsDependencyTreeNode[[dependencyNodes NutsDependencyTreeNode[[
getDependencyNodes()

```
####  descriptor  
return artifact descriptor
```

[read-only[NutsDescriptor descriptor NutsDescriptor getDescriptor()

```
####  effectiveDescriptor  
return artifact effective descriptor.  
this is an **optional** property.  
It must be requested (see `NutsSearchCommand#setEffective(boolean)` to be  
available).
```

[read-only[NutsDescriptor effectiveDescriptor NutsDescriptor getEffectiveDescriptor()

```
####  id  
artifact id
```

[read-only[NutsId id NutsId getId()

```
####  installInformation  
return artifact install information.
```

[read-only[NutsInstallInformation installInformation NutsInstallInformation getInstallInformation()

```
#### 📄 path  
return artifact content file path.  
this is an **optional** property. It must be requested (see ``  
NutsSearchCommand#setContent(boolean)``) to be available.
```

[read-only] Path path Path getPath()

```
#### 📄 repositoryName  
name of the repository providing this id.
```

[read-only] String repositoryName String getRepositoryName()

```
#### 📄 repositoryUuid  
id of the repository providing this id.
```

[read-only] String repositoryUuid String getRepositoryUuid()

```
#### 📄 setDependencies  
true if requested content
```

[read-only] boolean setDependencies boolean isSetDependencies()

```
#### 📄 setDependencyNodes  
true if requested content
```

[read-only] boolean setDependencyNodes boolean isSetDependencyNodes()

```
#### 📄 setEffectiveDescriptor  
true if requested effective descriptor
```

[read-only] boolean setEffectiveDescriptor boolean isSetEffectiveDescriptor()

```
#### 📄 type  
return artifact type
```

[read-only] NutsIdType type NutsIdType getType()

```
### ⚙ Instance Methods
#### ⚙ compareTo(other)
Compares this object with the specified definition for order.
This is equivalent to comparing subsequent ids.
```

```
int compareTo(NutsDefinition other)
```

```
other definition to compare with
```

```
## ☕ NutsExecutionContext
```

```
public net.vpc.app.nuts.NutsExecutionContext
```

```
execution context used in `NutsExecutorComponent` and
```

Instance Properties

📄 arguments

command arguments

```
[read-only] String[] arguments
String[] getArguments()
```

📄 commandName

command name

```
[read-only] String commandName
String getCommandName()
```

📄 cwd

current working directory

```
[read-only] String cwd  
String getcwd()
```

definition

command definition if any

```
[read-only] NutsDefinition definition  
NutsDefinition getDefinition()
```

env

execution environment

```
[read-only] Map<String, String> env  
Map<String, String> getEnv()
```

execSession

current session

```
[read-only] NutsSession execSession  
NutsSession getExecSession()
```

executionType

execution type

```
[read-only] NutsExecutionType executionType  
NutsExecutionType getExecutionType()
```

executorDescriptor

executor descriptor

```
[read-only] NutsArtifactCall executorDescriptor  
NutsArtifactCall getExecutorDescriptor()
```

executorOptions

executor options

```
[read-only] String[] executorOptions  
String[] getExecutorOptions()
```

executorProperties

executor properties

```
[read-only] Map<String, String> executorProperties  
Map<String, String> getExecutorProperties()
```

failFast

when true, any non 0 exited command will throw an Exception

```
[read-only] boolean failFast  
boolean isFailFast()
```

temporary

when true, the component is temporary and is not registered withing the workspace

```
[read-only] boolean temporary  
boolean isTemporary()
```

traceSession

```
[read-only] NutsSession traceSession  
NutsSession getTraceSession()
```

workspace

workspace

```
[read-only] NutsWorkspace workspace  
NutsWorkspace getWorkspace()
```

Instance Methods

workspace()

workspace

```
NutsWorkspace workspace()
```

NutsIOCompressAction

```
public net.vpc.app.nuts.NutsIOCompressAction
```

I/O Action that help monitored compress of one or multiple resource types. Default implementation should handle

Instance Properties

format

update format

```
[read-write] NutsIOCompressAction format  
String getFormat()  
NutsIOCompressAction setFormat(format)
```

logProgress

switch log progress flag to ` value` .

```
[read-write] NutsIOCompressAction logProgress  
boolean isLogProgress()  
NutsIOCompressAction setLogProgress(value)
```

progressMonitor

set progress monitor. Will create a singleton progress monitor factory

```
[write-only] NutsIOCompressAction progressMonitor  
NutsIOCompressAction setProgressMonitor(value)
```

progressMonitorFactory

set progress factory responsible of creating progress monitor

```
[read-write] NutsIOCompressAction progressMonitorFactory  
NutsProgressFactory getProgressMonitorFactory()  
NutsIOCompressAction setProgressMonitorFactory(value)
```

safe

switch safe copy flag to `value`

```
[read-write] NutsIOCompressAction safe  
boolean isSafe()  
NutsIOCompressAction setSafe(value)
```

session

update current session

```
[read-write] NutsIOCompressAction session  
NutsSession getSession()  
NutsIOCompressAction setSession(session)
```

📝 skipRoot

set skip root flag to `value`

```
[read-write] NutsIOCompressAction skipRoot  
boolean isSkipRoot()  
NutsIOCompressAction setSkipRoot(value)
```

📝 sources

sources to compress

```
[read-only] List<Object> sources  
List<Object> getSources()
```

📝 target

update target

```
[read-write] NutsIOCompressAction target  
Object getTarget()  
NutsIOCompressAction setTarget(target)
```

⚙ Instance Methods

⚙ addSource(source)

add source to compress

```
NutsIOCompressAction addSource(String source)
```

source

⚙ addSource(source)

add source to compress

```
NutsIOCompressAction addSource(InputStream source)
```

source

 **addSource**(source)

add source to compress

```
NutsIOCompressAction addSource(File source)
```

source

 **addSource**(source)

add source to compress

```
NutsIOCompressAction addSource(Path source)
```

source

 **addSource**(source)

add source to compress

```
NutsIOCompressAction addSource(URL source)
```

source

 **getFormatOption**(option)

return format option

```
Object getFormatOption(String option)
```

option name

⚙️ `logProgress()`

switch log progress flag to to true.

```
NutsIOCompressAction logProgress()
```

⚙️ `logProgress(value)`

switch log progress flag to `value` .

```
NutsIOCompressAction logProgress(boolean value)
```

value

⚙️ `progressMonitor(value)`

set progress monitor. Will create a singleton progress monitor factory

```
NutsIOCompressAction progressMonitor(NutsProgressMonitor value)
```

new value

⚙️ `progressMonitorFactory(value)`

set progress factory responsible of creating progress monitor

```
NutsIOCompressAction progressMonitorFactory(NutsProgressFactory value)
```

new value

⚙️ `run()`

run this Compress action

```
NutsIOCompressAction run()
```

⚙️ **safe()**

arm safe copy flag

```
NutsIOCompressAction safe()
```

⚙️ **safe(value)**

switch safe copy flag to `value`

```
NutsIOCompressAction safe(boolean value)
```

value

⚙️ **setFormatOption(option, value)**

update format option

```
NutsIOCompressAction setFormatOption(String option, Object value)
```

option name value

⚙️ **skipRoot()**

set skip root flag to `true`

```
NutsIOCompressAction skipRoot()
```

⚙️ **skipRoot(value)**

set skip root flag to `value`

```
NutsIOCompressAction skipRoot(boolean value)
```

new value

⚙️ [to\(target\)](#)

update target

```
NutsIOCompressAction to(OutputStream target)
```

target

⚙️ [to\(target\)](#)

update target

```
NutsIOCompressAction to(String target)
```

target

⚙️ [to\(target\)](#)

update target

```
NutsIOCompressAction to(Path target)
```

target

⚙️ [to\(target\)](#)

update target

```
NutsIOCompressAction to(File target)
```

target

⚙️ [to\(target\)](#)

update target

```
NutsIOCompressAction to(Object target)
```

target

⌚ NutsIdLocationBuilder

```
public net.vpc.app.nuts.NutsIdLocationBuilder
```

Mutable IdLocation class that helps creating instance of immutable `NutsIdLocation` . Instances of `NutsIdLocation` are used in `NutsDescriptor` (see `NutsDescriptor#getLocations()`) ===== ? Instance Properties

📝 classifier

update location classifier

```
[read-write] NutsIdLocationBuilder classifier  
String getClassifier()  
NutsIdLocationBuilder setClassifier(value)
```

📝 region

update location region

```
[read-write] NutsIdLocationBuilder region  
String getRegion()  
NutsIdLocationBuilder setRegion(value)
```

📝 url

update location url

```
[read-write] NutsIdLocationBuilder url  
String getUrl()  
NutsIdLocationBuilder setUrl(value)
```

⚙ Instance Methods

⚙ `build()`

create new instance of `NutsIdLocation` initialized with this builder values.

`NutsIdLocation build()`

⚙ `classifier(value)`

update location classifier

`NutsIdLocationBuilder classifier(String value)`

location classifier

⚙ `clear()`

clear this instance (set null/default all properties)

`NutsIdLocationBuilder clear()`

⚙ `region(value)`

update location region

`NutsIdLocationBuilder region(String value)`

location region

⚙ `set(value)`

update all attributes, copy from `value` instance

`NutsIdLocationBuilder set(NutsIdLocationBuilder value)`

instance to copy from

 **set(value)**

update all attributes, copy from `value` instance

```
NutsIdLocationBuilder set(NutsIdLocation value)
```

instance to copy from

 **url(value)**

update location url

```
NutsIdLocationBuilder url(String value)
```

location url

 **NutsIndexStore**

```
public net.vpc.app.nuts.NutsIndexStore
```

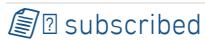
Classes implementations of `NutsIndexStore` handle indexing of repositories to enable faster search.

 **Instance Properties**

  **enabled**

enable or disable of index

```
[read-write] NutsIndexStore enabled  
boolean isEnabled()  
NutsIndexStore setEnabled(enabled)
```



return true if the current repository is registered

```
[read-only] boolean subscribed  
boolean isSubscribed()
```

⚙ Instance Methods

⚙ **enabled()**

enable index

```
NutsIndexStore enabled()
```

⚙ **enabled(enabled)**

enable or disable of index

```
NutsIndexStore enabled(boolean enabled)
```

new value

⚙ **invalidate(id)**

invalidate the artifact from the index

```
NutsIndexStore invalidate(NutsId id)
```

id to invalidate

⚙ **revalidate(id)**

invalidate the artifact from the index and re-index it

```
NutsIndexStore revalidate(NutsId id)
```

id to re-index

```
⚙️ search(filter, session)
```

search all artifacts matching the given filter

```
Iterator<NutsId> search(NutsIdFilter filter, NutsSession session)
```

filter or null for all current session

```
⚙️ searchVersions(id, session)
```

search all versions of the given artifact

```
Iterator<NutsId> searchVersions(NutsId id, NutsSession session)
```

artifact to search for current session

```
⚙️ subscribe()
```

subscribe the current repository so the indexing is processed.

```
NutsIndexStore subscribe()
```

```
⚙️ unsubscribe()
```

unsubscribe the current repository so that the indexing is disabled and the index is removed.

```
NutsIndexStore unsubscribe()
```

👉 [NutsIndexStoreFactory](#)

```
public net.vpc.app.nuts.NutsIndexStoreFactory
```

Index Store Factory responsible of creating stores for a given repository

⚙ Instance Methods

⚙ `createIndexStore(repository)`

create a new index store implementation or null if not supported

```
NutsIndexStore createIndexStore(NutsRepository repository)
```

repository to create the index store to

📥 NutsInputStreamTransparentAdapter

```
public net.vpc.app.nuts.NutsInputStreamTransparentAdapter
```

Interface to enable marking system streams. When creating new processes nuts will dereference NutsInputStreamTransparentAdapter to check if the InputStream is a system io. In that case nuts will "inherit" input stream

⚙ Instance Methods

⚙ `baseInputStream()`

de-referenced stream

```
InputStream baseInputStream()
```

📥 NutsInstallCommand

```
public net.vpc.app.nuts.NutsInstallCommand
```

Command for installing artifacts

Instance Properties

args

return all arguments to pass to the install command

```
[read-only] String[] args  
String[] getArgs()
```

companions

if true update companions

```
[read-write] NutsInstallCommand companions  
boolean isCompanions()  
NutsInstallCommand setCompanions(value)
```

defaultVersion

set default version flag. when true, the installed version will be defined as default

```
[read-write] NutsInstallCommand defaultVersion  
boolean isDefaultVersion()  
NutsInstallCommand setDefaultVersion(defaultVersion)
```

ids

return all ids to install

```
[read-only] NutsId[] ids  
NutsId[] getIds()
```

installed

if true reinstall installed artifacts

```
[read-write] NutsInstallCommand installed  
boolean isInstalled()  
NutsInstallCommand setInstalled(value)
```

📄 result

execute installation and return result.

```
[read-only] NutsResultList<NutsDefinition> result  
NutsResultList<NutsDefinition> getResult()
```

@update session

update session

```
[write-only] NutsInstallCommand session  
NutsInstallCommand setSession(session)
```

⚙️ Instance Methods

⚙️ addArg(arg)

add argument to pass to the install command

```
NutsInstallCommand addArg(String arg)
```

argument

⚙️ addArgs(args)

add arguments to pass to the install command

```
NutsInstallCommand addArgs(Collection<String> args)
```

argument

 **addArgs(args)**

add arguments to pass to the install command

```
NutsInstallCommand addArgs(String args)
```

argument

 **addId(id)**

add artifact id to install

```
NutsInstallCommand addId(NutsId id)
```

id to install

 **addId(id)**

add artifact id to install

```
NutsInstallCommand addId(String id)
```

id to install

 **addIds(ids)**

add artifact ids to install

```
NutsInstallCommand addIds(NutsId ids)
```

ids to install

 **addIds(ids)**

add artifact ids to install

```
NutsInstallCommand addIds(String ids)
```

ids to install

⚙️ `arg(arg)`

add argument to pass to the install command

```
NutsInstallCommand arg(String arg)
```

argument

⚙️ `args(args)`

add arguments to pass to the install command

```
NutsInstallCommand args(Collection<String> args)
```

argument

⚙️ `args(args)`

add arguments to pass to the install command

```
NutsInstallCommand args(String args)
```

argument

⚙️ `clearArgs()`

clear all arguments to pass to the install command

```
NutsInstallCommand clearArgs()
```

⚙️ `clearIds()`

clear ids to install

```
NutsInstallCommand clearIds()
```

⚙️ `companions()`

update companions

```
NutsInstallCommand companions()
```

⚙️ `companions(value)`

if true update companions

```
NutsInstallCommand companions(boolean value)
```

flag

⚙️ `configure(skipUnsupported, args)`

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsInstallCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ `copySession()`

copy session

```
NutsInstallCommand copySession()
```

⚙️ defaultVersion()

set default version flag. the installed version will be defined as default.

```
NutsInstallCommand defaultVersion()
```

⚙️ defaultVersion(defaultVersion)

set default version flag. when true, the installed version will be defined as default

```
NutsInstallCommand defaultVersion(boolean defaultVersion)
```

when true, the installed version will be defined as default

⚙️ id(id)

add artifact id to install

```
NutsInstallCommand id(NutsId id)
```

id to install

⚙️ id(id)

add artifact id to install

```
NutsInstallCommand id(String id)
```

id to install

⚙️ ids(ids)

add artifact ids to install

```
NutsInstallCommand ids(NutsId ids)
```

id to install

 **ids(ids)**

add artifact ids to install

```
NutsInstallCommand ids(String ids)
```

id to install

 **installed()**

reinstall installed artifacts

```
NutsInstallCommand installed()
```

 **installed(value)**

if true reinstall installed artifacts

```
NutsInstallCommand installed(boolean value)
```

flag

 **removeld(id)**

remove artifact id to install

```
NutsInstallCommand removeId(NutsId id)
```

id to install

 **removeld(id)**

remove artifact id to install

```
NutsInstallCommand removeId(String id)
```

id to install

 **run()**

execute the command and return this instance

```
NutsInstallCommand run()
```

 **NutsInstallEvent**

```
public net.vpc.app.nuts.NutsInstallEvent
```

Event describing installation of an artifact

 **Instance Properties**

  **definition**

return artifact definition

```
[read-only] NutsDefinition definition  
NutsDefinition getDefinition()
```

  **force**

return true if installation was forced

```
[read-only] boolean force  
boolean isForce()
```

  **session**

return current session

```
[read-only] NutsSession session  
NutsSession getSession()
```

[workspace](#)

current workspace

```
[read-only] NutsWorkspace workspace  
NutsWorkspace getWorkspace()
```

[NutsInstallInformation](#)

```
public net.vpc.app.nuts.NutsInstallInformation
```

Information about installed artifact

[Instance Properties](#)

[defaultVersion](#)

true when the installed artifact is default version

```
[read-only] boolean defaultVersion  
boolean isDefaultVersion()
```

[id](#)

installation date

```
[read-only] NutsId id  
NutsId getId()
```

[installDate](#)

installation date

```
[read-only] Instant installDate  
Instant getInstallDate()
```

installFolder

installation formation path.

```
[read-only] Path installFolder  
Path getInstallFolder()
```

installStatus

return install status

```
[read-only] NutsInstallStatus installStatus  
NutsInstallStatus getInstallStatus()
```

installUser

return the user responsible of the installation

```
[read-only] String installUser  
String getInstallUser()
```

installedOrIncluded

return true if installed primary or dependency

```
[read-only] boolean installedOrIncluded  
boolean isInstalledOrIncluded()
```

justInstalled

true if the installation just occurred in the very last operation

```
[read-only] boolean justInstalled  
boolean isJustInstalled()
```

justReinstalled

true if the re-installation just occurred in the very last operation

```
[read-only] boolean justReInstalled  
boolean isJustReInstalled()
```

sourceRepositoryName

```
[read-only] String sourceRepositoryName  
String getSourceRepositoryName()
```

sourceRepositoryUUID

```
[read-only] String sourceRepositoryUUID  
String getSourceRepositoryUUID()
```

NutsListener

```
public net.vpc.app.nuts.NutsListener
```

Anchor interface for all Nuts Listeners.

NutsMapListener

```
public net.vpc.app.nuts.NutsMapListener
```

Map Listener to catch updates

⚙ Instance Methods

⚙ entryAdded(key, value)

Invoked when item added

```
void entryAdded(K key, V value)
```

key value

⚙ entryRemoved(key, value)

Invoked when item removed

```
void entryRemoved(K key, V value)
```

key value

⚙ entryUpdated(key, newValue, oldValue)

Invoked when item updated

```
void entryUpdated(K key, V newValue, V oldValue)
```

key new value old value

☕ NutsProcessInfo

```
public net.vpc.app.nuts.NutsProcessInfo
```

System Process Information

ⓘ Instance Properties

commandLine

Process command line

```
[read-only] String commandLine  
String getCommandLine()
```

name

Process Name. This should represent Fully Qualified Java Main Class Name for java processes.

```
[read-only] String name  
String getName()
```

pid

Process Id in string representation

```
[read-only] String pid  
String getPid()
```

title

Process Title / Window Title if available

```
[read-only] String title  
String getTitle()
```

NutsProgressFactory

```
public net.vpc.app.nuts.NutsProgressFactory
```

NutsProgressFactory is responsible of creating instances of `NutsProgressMonitor` ===== 

Instance Methods

⚙️ `create(source, sourceOrigin, session)`

create a new instance of ` NutsProgressMonitor`

```
NutsProgressMonitor create(Object source, Object sourceOrigin, NutsSession session)
```

source object of the progress. This may be the File for instance source origin object of the progress. This may be the NutsId for instance workspace session

☕ `NutsProgressMonitor`

```
public net.vpc.app.nuts.NutsProgressMonitor
```

Monitor handles events from copy, compress and delete actions

⚙️ Instance Methods

⚙️ `onComplete(event)`

called when the action terminates

```
void onComplete(NutsProgressEvent event)
```

event

⚙️ `onProgress(event)`

called when the action does a step forward and return true if the progress was handled of false otherwise.

```
boolean onProgress(NutsProgressEvent event)
```

event

⚙️ onStart(event)

called when the action starts

```
void onStart(NutsProgressEvent event)
```

event

📢 NutsQuestion

```
public net.vpc.app.nuts.NutsQuestion
```

Question is helpful object that permits user interaction by reading a typed object from standard input or an equivalent input system.

⌚ Instance Properties

📝⌚ acceptedValues

```
[read-write] NutsQuestion<T> acceptedValues  
Object[] getAcceptedValues()  
NutsQuestion<T> setAcceptedValues(acceptedValues)
```

📝⌚ booleanValue

equivalent to (Boolean) getValue() as type dereferencing may cause some troubles

```
[read-only] Boolean booleanValue  
Boolean getBooleanValue()
```

📝⌚ defaultValue

```
[read-write] NutsQuestion<T> defaultValue  
T getDefaultValue()  
NutsQuestion<T> setDefaultvalue(defaultValue)
```

 **format**

```
[read-write] NutsQuestion<T> format  
NutsQuestionFormat<T> getFormat()  
NutsQuestion<T> setFormat(format)
```

 **hintMessage**

```
[read-only] String hintMessage  
String getHintMessage()
```

 **hintMessageParameters**

```
[read-only] Object[] hintMessageParameters  
Object[] getHintMessageParameters()
```

 **message**

```
[read-only] String message  
String getMessage()
```

 **messageParameters**

```
[read-only] Object[] messageParameters  
Object[] getMessageParameters()
```

 **parser**

```
[read-write] NutsQuestion<T> parser  
NutsQuestionParser<T> getParser()  
NutsQuestion<T> setParser(parser)
```

session

```
[read-write] NutsQuestion<T> session
NutsSession getSession()
NutsQuestion<T> setSession(session)
```

validator

```
[read-write] NutsQuestion<T> validator
NutsQuestionValidator<T> getValidator()
NutsQuestion<T> setValidator(validator)
```

value

```
[read-only] T value
T getValue()
```

valueType

```
[read-write] NutsQuestion<T> valueType
Class<T> getValueType()
NutsQuestion<T> setValueType(valueType)
```

Instance Methods

acceptedValues(acceptedValues)

```
NutsQuestion<T> acceptedValues(Object[] acceptedValues)
```

null

configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsQuestion<T> configure(boolean skipUnsupported, String args)
```

null argument to configure with

⚙ defaultValue(defaultValue)

```
NutsQuestion<T> defaultValue(T defaultValue)
```

null

⚙ forBoolean(msg, params)

```
NutsQuestion<Boolean> forBoolean(String msg, Object params)
```

null null

⚙ forDouble(msg, params)

```
NutsQuestion<Double> forDouble(String msg, Object params)
```

null null

⚙ forEnum(enumType, msg, params)

```
NutsQuestion<K> forEnum(Class<K> enumType, String msg, Object params)
```

null null null

⚙ forFloat(msg, params)

```
NutsQuestion<Float> forFloat(String msg, Object params)
```

null null

⚙️ forInteger(msg, params)

```
NutsQuestion<Integer> forInteger(String msg, Object params)
```

null null

⚙️ forLong(msg, params)

```
NutsQuestion<Long> forLong(String msg, Object params)
```

null null

⚙️ forPassword(msg, params)

```
NutsQuestion<char[]> forPassword(String msg, Object params)
```

null null

⚙️ forString(msg, params)

```
NutsQuestion<String> forString(String msg, Object params)
```

null null

⚙️ format(format)

```
NutsQuestion<T> format(NutsQuestionFormat<T> format)
```

null

⚙️ hintMessage(message, messageParameters)

```
NutsQuestion<T> hintMessage(String message, Object messageParameters)
```

null null

⚙️ **message(message, messageParameters)**

```
NutsQuestion<T> message(String message, Object messageParameters)
```

null null

⚙️ **parser(parser)**

```
NutsQuestion<T> parser(NutsQuestionParser<T> parser)
```

null

⚙️ **run()**

```
NutsQuestion<T> run()
```

⚙️ **setHintMessage(message, messageParameters)**

```
NutsQuestion<T> setHintMessage(String message, Object messageParameters)
```

null null

⚙️ **setMessage(message, messageParameters)**

```
NutsQuestion<T> setMessage(String message, Object messageParameters)
```

null null

⚙️ **validator(validator)**

```
NutsQuestion<T> validator(NutsQuestionValidator<T> validator)
```

null

⚙️ valueType(valueType)

```
NutsQuestion<T> valueType(Class<T> valueType)
```

null

☕ NutsQuestionParser

```
public net.vpc.app.nuts.NutsQuestionParser
```

⚙️ Instance Methods

⚙️ parse(response, defaultValue, question)

```
T parse(Object response, T defaultValue, NutsQuestion<T> question)
```

null null null

☕ NutsQuestionValidator

```
public net.vpc.app.nuts.NutsQuestionValidator
```

⚙️ Instance Methods

⚙️ validate(value, question)

```
T validate(T value, NutsQuestion<T> question)
```

null null

☕ NutsRepository

```
public net.vpc.app.nuts.NutsRepository
```

Nuts repository manages a set of packages

[Instance Fields](#)

[SPEED_FAST](#)

```
int SPEED_FAST = 10000
```

[SPEED_FASTER](#)

```
int SPEED_FASTER = 100000
```

[SPEED_FASTEST](#)

```
int SPEED_FASTEST = 1000000
```

[SPEED_SLOW](#)

```
int SPEED_SLOW = 1000
```

[SPEED_SLOWER](#)

```
int SPEED_SLOWER = 100
```

[SPEED_SLOWEST](#)

```
int SPEED_SLOWEST = 10
```

[Instance Properties](#)

[name](#)

return repository name. equivalent to config().name()

```
[read-only] String name  
String getName()
```

parentRepository

return parent repository or null

```
[read-only] NutsRepository parentRepository  
NutsRepository getParentRepository()
```

repositoryListeners

Repository Listeners

```
[read-only] NutsRepositoryListener[] repositoryListeners  
NutsRepositoryListener[] getRepositoryListeners()
```

repositoryType

return repository type

```
[read-only] String repositoryType  
String getRepositoryType()
```

userPropertyListeners

return array of registered user properties listeners

```
[read-only] NutsMapListener<String, Object>[] userPropertyListeners  
NutsMapListener<String, Object>[] getUserPropertyListeners()
```

uuid

return repository unique identifier

```
[read-only] String uuid  
String getUuid()
```

📄 workspace

return parent workspace

```
[read-only] NutsWorkspace workspace  
NutsWorkspace getWorkspace()
```

⚙️ Instance Methods

⚙️ addRepositoryListener(listener)

add repository listener

```
void addRepositoryListener(NutsRepositoryListener listener)
```

listener

⚙️ addUserPropertyListener(listener)

add listener to user properties

```
void addUserPropertyListener(NutsMapListener<String, Object> listener)
```

listener

⚙️ config()

return repository configuration manager

```
NutsRepositoryConfigManager config()
```

⚙️ `deploy()`

create deploy command

```
NutsDeployRepositoryCommand deploy()
```

⚙️ `fetchContent()`

create fetchContent command

```
NutsFetchContentRepositoryCommand fetchContent()
```

⚙️ `fetchDescriptor()`

create fetchDescriptor command

```
NutsFetchDescriptorRepositoryCommand fetchDescriptor()
```

⚙️ `name()`

return repository name. equivalent to config().name()

```
String name()
```

⚙️ `parentRepository()`

return parent repository or null

```
NutsRepository parentRepository()
```

⚙️ `push()`

create push command

NutsPushRepositoryCommand **push()**

⚙️ **removeRepositoryListener(listener)**

remove repository listener

void **removeRepositoryListener**(NutsRepositoryListener listener)

listener

⚙️ **removeUserPropertyListener(listener)**

remove listener from user properties

void **removeUserPropertyListener**(NutsMapListener<String, Object> listener)

listener

⚙️ **repositoryType()**

return repository type

String **repositoryType()**

⚙️ **search()**

create search command

NutsSearchRepositoryCommand **search()**

⚙️ **searchVersions()**

create searchVersions command

```
NutsSearchVersionsRepositoryCommand searchVersions()
```

⚙️ **security()**

return repository security manager

```
NutsRepositorySecurityManager security()
```

⚙️ **undeploy()**

create undeploy command

```
NutsRepositoryUndeployCommand undeploy()
```

⚙️ **updateStatistics()**

create update statistics command

```
NutsUpdateRepositoryStatisticsCommand updateStatistics()
```

⚙️ **userProperties()**

return mutable instance of user properties

```
Map<String, Object> userProperties()
```

⚙️ **uuid()**

return repository unique identifier

```
String uuid()
```

⚙️ workspace()

return parent workspace

```
NutsWorkspace workspace()
```

☕ NutsRepositoryFilter

```
public net.vpc.app.nuts.NutsRepositoryFilter
```

Created by vpc on 1/5/17.

⚙️ Instance Methods

⚙️ accept(repository)

```
boolean accept(NutsRepository repository)
```

null

☕ NutsResultList

```
public net.vpc.app.nuts.NutsResultList
```

Find Result items from find command

⚙️ Instance Methods

⚙️ count()

return elements count of this result.

consumes the result and returns the number of elements consumed. Calling this method twice will result in unexpected behavior (may return 0 as the result is already consumed or throw an Exception)

```
long count()
```

⚙ first()

return the first value or null if none found.

Calling this method twice will result in unexpected behavior (may return an incorrect value such as null as the result is already consumed or throw an Exception)

```
T first()
```

⚙ list()

return result as a java.util.List .

consumes the result and returns a list Calling this method twice will result in unexpected behavior (may return an empty list as the result is already consumed or throw an Exception)

```
List<T> list()
```

⚙ required()

return the first value or NutsNotFoundException if not found.

Calling this method twice will result in unexpected behavior (may return an incorrect value such as null as the result is already consumed or throw an Exception)

```
T required()
```

⚙ singleton()

return the first value while checking that there are no more elements.

Calling this method twice will result in unexpected behavior (may return an incorrect value such as null as the result is already consumed or throw an Exception)

T **singleton()**

⚙ stream()

return result as a java.util.stream.Stream .

Calling this method twice will result in unexpected behavior (may return 0 as the result is already consumed or throw an Exception)

Stream<T> **stream()**

☕ NutsSearchId

public net.vpc.app.nuts.NutsSearchId

Search id defines a uniform interface to ids, versions and descriptors

⚙ Instance Methods

⚙ **getDescriptor(session)**

return descriptor

NutsDescriptor **getDescriptor(NutsSession session)**

session

⚙ **getId(session)**

return id

NutsId **getId(NutsSession session)**

session

⚙️ getVersion(session)

return version

```
NutsVersion getVersion(NutsSession session)
```

session

〝 NutsSearchIdFilter

```
public net.vpc.app.nuts.NutsSearchIdFilter
```

SearchId Filter.

⚙️ Instance Methods

⚙️ acceptSearchId(sid, session)

true if search id is accepted

```
boolean acceptSearchId(NutsSearchId sid, NutsSession session)
```

search id session

〝 NutsSession

```
public net.vpc.app.nuts.NutsSession
```

session is context defining common command options and parameters.

ⓘ Instance Properties

📝 ⓘ ask

equivalent to `setConfirm(enable?ASK:null)`

```
[read-write] NutsSession ask  
boolean isAsk()  
NutsSession setAsk(enable)
```

cached

use cache

```
[read-write] NutsSession cached  
boolean isCached()  
NutsSession setCached(value)
```

confirm

set confirm mode.

```
[read-write] NutsSession confirm  
NutsConfirmationMode getConfirm()  
NutsSession setConfirm(confirm)
```

fetchStrategy

change fetch strategy

```
[read-write] NutsSession fetchStrategy  
NutsFetchStrategy getFetchStrategy()  
NutsSession setFetchStrategy(mode)
```

force

change force flag value. some operations may require user confirmation before performing critical operations such as overriding existing values, deleting sensitive information ; in such cases, arming force flag will provide an implicit confirmation.

```
[read-write] NutsSession force  
boolean isForce()  
NutsSession setForce(enable)
```

 indexed

use index

```
[read-write] NutsSession indexed  
boolean isIndexed()  
NutsSession setIndexed(value)
```

 iterableFormat

set iterable output format

```
[read-write] NutsSession iterableFormat  
NutsIterableFormat getIterableFormat()  
NutsSession setIterableFormat(value)
```

 iterableOut

true if iterable format is armed. equivalent to `getIterableFormat() != null`

```
[read-only] boolean iterableOut  
boolean isIterableOut()
```

 iterableOutput

return iterable output

```
[read-only] NutsIterableOutput iterableOutput  
NutsIterableOutput getIterableOutput()
```

iterableTrace

true if iterable format and trace flag are armed. equivalent to `isTrace()` && isIterableOut() ``

```
[read-only] boolean iterableTrace  
boolean isIterableTrace()
```

listeners

return all registered listeners.

```
[read-only] NutsListener[] listeners  
NutsListener[] getListeners()
```

no

change no flag value. some operations may require user confirmation before performing critical operations such as overriding existing values, deleting sensitive information ; in such cases, arming no flag will provide an implicit negative confirmation.

```
[read-write] NutsSession no  
boolean isNo()  
NutsSession setNo(enable)
```

outputFormat

set output format

```
[read-write] NutsSession outputFormat  
NutsOutputFormat getOutputFormat()  
NutsSession setOutputFormat(outputFormat)
```

outputFormatOptions

set output format options (clear and add)

```
[read-write] NutsSession outputFormatOptions  
String[] getOutputFormatOptions()  
NutsSession setOutputFormatOptions(options)
```

plainOut

true if NON iterable and plain format are armed.

```
[read-only] boolean plainOut  
boolean isPlainOut()
```

plainTrace

true if non iterable and plain formats along with trace flag are armed. equivalent to `isTrace()` && !isIterableOut() && getOutputFormat() == NutsOutputFormat.PLAIN ``

```
[read-only] boolean plainTrace  
boolean isPlainTrace()
```

progressOptions

change progress options

```
[read-write] NutsSession progressOptions  
String getProgressOptions()  
NutsSession setProgressOptions(progressOptions)
```

properties

add session properties

```
[read-write] NutsSession properties  
Map<String, Object> getProperties()  
NutsSession setProperties(properties)
```

structuredOut

true if NON iterable and NON plain formats are armed. equivalent to ` !isIterableOut()` && getOutputFormat() != NutsOutputFormat.PLAIN ``

```
[read-only] boolean structuredOut  
boolean isStructuredOut()
```

structuredTrace

true if NON iterable and NON plain formats along with trace flag are armed. equivalent to ` isTrace()` && !isIterableOut() && getOutputFormat() == NutsOutputFormat.PLAIN ``

```
[read-only] boolean structuredTrace  
boolean isStructuredTrace()
```

terminal

set session terminal

```
[read-write] NutsSession terminal  
NutsSessionTerminal getTerminal()  
NutsSession setTerminal(terminal)
```

trace

change trace flag value. When true, operations are invited to print to output stream information about processing. Output may be in different formats according to ` #getOutputFormat()` and ` #getIterableFormat()`

```
[read-write] NutsSession trace  
boolean isTrace()  
NutsSession setTrace(trace)
```

transitive

consider transitive repositories

```
[read-write] NutsSession transitive  
boolean isTransitive()  
NutsSession setTransitive(value)
```

workspace

current workspace

```
[read-only] NutsWorkspace workspace  
NutsWorkspace getWorkspace()
```

yes

change YES flag value. some operations may require user confirmation before performing critical operations such as overriding existing values, deleting sensitive information ; in such cases, arming yes flag will provide an implicit confirmation.

```
[read-write] NutsSession yes  
boolean isYes()  
NutsSession setYes(enable)
```

Instance Methods

addListener(listener)

add session listener. supported listeners are instances of:

- `NutsWorkspaceListener`
- `NutsInstallListener`
- `NutsMapListener`
- `NutsRepositoryListener`

```
NutsSession addListener(NutsListener listener)
```

listener

⚙ addOutputFormatOptions(options)

add output format options

```
NutsSession addOutputFormatOptions(String options)
```

output format options.

⚙ ask()

equivalent to `setAsk(true)`

```
NutsSession ask()
```

⚙ confirm(confirm)

set confirm mode.

```
NutsSession confirm(NutsConfirmationMode confirm)
```

confirm type.

⚙ copy()

return new instance copy of `this` session

```
NutsSession copy()
```

⚙ copyFrom(other)

copy into this instance from the given value

```
NutsSession copyFrom(NutsSession other)
```

other session to copy from

⚙️ `err()`

current error stream

```
PrintStream err()
```

⚙️ `fetchAnyWhere()`

change fetch strategy to ANYWHERE

```
NutsSession fetchAnyWhere()
```

⚙️ `fetchOffline()`

change fetch strategy to OFFLINE

```
NutsSession fetchOffline()
```

⚙️ `fetchOnline()`

change fetch strategy to ONLINE

```
NutsSession fetchOnline()
```

⚙️ `fetchRemote()`

change fetch strategy to REMOTE

```
NutsSession fetchRemote()
```

⚙️ `fetchStrategy(mode)`

change fetch strategy

```
NutsSession fetchStrategy(NutsFetchStrategy mode)
```

new strategy or null

⚙️ `formatObject(any)`

This is a helper method to create an Object format initialized with this session instance and the given object to print. `thisSession.getWorkspace().object().setSession(thisSession).value(any)`

Using this method is recommended to print objects to default format (json, xml,...)

```
NutsObjectFormat formatObject(Object any)
```

any object to print in the configured/default format

⚙️ `getListeners(type)`

return registered listeners for the given type. Supported types are :

- `NutsWorkspaceListener`
- `NutsInstallListener`
- `NutsMapListener`
- `NutsRepositoryListener`

```
T[] getListeners(Class<T> type)
```

listener type class

⚙️ `getOutputFormat(defaultValue)`

return current Output Format or `defaultValue` if null

```
NutsOutputFormat getOutputFormat(NutsOutputFormat defaultValue)
```

value when Output Format is not set

 **getProperty(key)**

return property value or null

```
Object getProperty(String key)
```

property key

 **json()**

set json output format

```
NutsSession json()
```

 **no()**

equivalent to `setNo(true)`

```
NutsSession no()
```

 **no(enable)**

equivalent to `setNo(enable)`

```
NutsSession no(boolean enable)
```

new value

 **out()**

current output stream

```
PrintStream out()
```

⚙️ **plain()**

set plain text (default) output format

```
NutsSession plain()
```

⚙️ **props()**

set properties output format

```
NutsSession props()
```

⚙️ **removeListener(listener)**

remove session listener. supported listeners are instances of:

- `NutsWorkspaceListener`
- `NutsInstallListener`
- `NutsMapListener`
- `NutsRepositoryListener`

```
NutsSession removeListener(NutsListener listener)
```

listener

⚙️ **setProperty(key, value)**

set session property

```
NutsSession setProperty(String key, Object value)
```

property key property value

⚙️ **setSilent()**

equivalent to `setTrace(false)`

NutsSession **setSilent()**

⚙️ **table()**

set table output format

NutsSession **table()**

⚙️ **terminal()**

current terminal

NutsSessionTerminal **terminal()**

⚙️ **tree()**

set tree output format

NutsSession **tree()**

⚙️ **workspace()**

current workspace

NutsWorkspace **workspace()**

⚙️ **xml()**

set xml output format

NutsSession [xml\(\)](#)

⚙️ [yes\(\)](#)

equivalent to `setYes(true)`

NutsSession [yes\(\)](#)

⚙️ [yes\(enable\)](#)

equivalent to `setYes(enable)`

NutsSession [yes\(boolean enable\)](#)

new value

☕ NutsSessionTerminal

public net.vpc.app.nuts.NutsSessionTerminal

Created by vpc on 2/20/17.

🔗 Instance Properties

🔗 err

[write-only] void err
void [setErr\(out\)](#)

🔗 in

[write-only] void in
void [setIn\(in\)](#)

 out

```
[write-only] void out  
void setOut(out)
```

 parent

```
[read-write] void parent  
NutsTerminalBase getParent()  
void setParent(parent)
```

 Instance Methods

 copy()

```
NutsSessionTerminal copy()
```

 NutsTokenFilter

```
public net.vpc.app.nuts.NutsTokenFilter
```

 Instance Properties

 blank

```
[read-only] boolean blank  
boolean isBlank()
```

 null

```
[read-only] boolean null  
boolean isNull()
```

⚙ Instance Methods

⚙ contains(substring)

```
boolean contains(String substring)
```

null

⚙ like(pattern)

```
boolean like(String pattern)
```

null

⚙ matches(pattern)

```
boolean matches(String pattern)
```

null

☕ NutsVersionInterval

```
public net.vpc.app.nuts.NutsVersionInterval
```

Created by vpc on 2/1/17.

⌚ Instance Properties

⌚ fixedValue

```
[read-only] boolean fixedValue  
boolean isFixedValue()
```

includeLowerBound

```
[read-only] boolean includeLowerBound  
boolean isIncludeLowerBound()
```

includeUpperBound

```
[read-only] boolean includeUpperBound  
boolean isIncludeUpperBound()
```

lowerBound

```
[read-only] String lowerBound  
String getLowerBound()
```

upperBound

```
[read-only] String upperBound  
String getUpperBound()
```

Instance Methods

acceptVersion(version)

```
boolean acceptVersion(NutsVersion version)
```

null

NutsWorkspace

```
public net.vpc.app.nuts.NutsWorkspace
```

Created by vpc on 1/5/17.

Instance Properties

installListeners

```
[read-only] NutsInstallListener[] installListeners  
NutsInstallListener[] getInstallListeners()
```

name

Workspace name

```
[read-only] String name  
String getName()
```

repositoryListeners

```
[read-only] NutsRepositoryListener[] repositoryListeners  
NutsRepositoryListener[] getRepositoryListeners()
```

userPropertyListeners

```
[read-only] NutsMapListener<String, Object>[] userPropertyListeners  
NutsMapListener<String, Object>[] getUserPropertyListeners()
```

uuid

Workspace identifier, guaranteed to be unique cross machines

```
[read-only] String uuid  
String getUuid()
```

workspaceListeners

```
[read-only] NutsWorkspaceListener[] workspaceListeners  
NutsWorkspaceListener[] getWorkspaceListeners()
```

⚙ Instance Methods

⚙ addInstallListener(listener)

```
void addInstallListener(NutsInstallListener listener)
```

null

⚙ addRepositoryListener(listener)

```
void addRepositoryListener(NutsRepositoryListener listener)
```

null

⚙ addUserPropertyListener(listener)

```
void addUserPropertyListener(NutsMapListener<String, Object> listener)
```

null

⚙ addWorkspaceListener(listener)

```
void addWorkspaceListener(NutsWorkspaceListener listener)
```

null

⚙ commandLine()

```
NutsCommandLineFormat commandLine()
```

⚙ config()

NutsWorkspaceConfigManager **config()**

⚙ createSession()

NutsSession **createSession()**

⚙ dependency()

create dependency format instance

NutsDependencyFormat **dependency()**

⚙ deploy()

NutsDeployCommand **deploy()**

⚙ descriptor()

create descriptor format instance

NutsDescriptorFormat **descriptor()**

⚙ element()

create element format instance

NutsElementFormat **element()**

⚙ exec()

NutsExecCommand **exec()**

⚙️ **extensions()**

```
NutsWorkspaceExtensionManager extensions()
```

⚙️ **fetch()**

```
NutsFetchCommand fetch()
```

⚙️ **id()**

create id format instance

```
NutsIdFormat id()
```

⚙️ **info()**

create info format instance

```
NutsInfoFormat info()
```

⚙️ **install()**

```
NutsInstallCommand install()
```

⚙️ **io()**

```
NutsIOManager io()
```

⚙️ **iter()**

create iterable format instance

```
NutsIterableOutput iter()
```

⚙ json()

create json format instance

```
NutsJsonFormat json()
```

⚙ log()

```
NutsLogManager log()
```

⚙ name()

equivalent to `#getName()`

```
String name()
```

⚙ object()

create object format instance

```
NutsObjectFormat object()
```

⚙ props()

create properties format instance

```
NutsPropertiesFormat props()
```

⚙ push()

```
NutsPushCommand push()
```

⚙️ **removeInstallListener(listener)**

```
void removeInstallListener(NutsInstallListener listener)
```

null

⚙️ **removeRepositoryListener(listener)**

```
void removeRepositoryListener(NutsRepositoryListener listener)
```

null

⚙️ **removeUserPropertyListener(listener)**

```
void removeUserPropertyListener(NutsMapListener<String, Object> listener)
```

null

⚙️ **removeWorkspaceListener(listener)**

```
void removeWorkspaceListener(NutsWorkspaceListener listener)
```

null

⚙️ **search()**

```
NutsSearchCommand search()
```

⚙️ **security()**

```
NutsWorkspaceSecurityManager security()
```

⚙ str()

create string format instance

NutsStringFormat str()

⚙ table()

create table format instance

NutsTableFormat table()

⚙ tree()

create tree format instance

NutsTreeFormat tree()

⚙ undeploy()

NutsUndeployCommand undeploy()

⚙ uninstall()

NutsUninstallCommand uninstall()

⚙ update()

NutsUpdateCommand update()

⚙ updateStatistics()

NutsUpdateStatisticsCommand updateStatistics()

⚙️ **userProperties()**

```
Map<String, Object> userProperties()
```

⚙️ **uuid()**

equivalent to `#getUuid()`

```
String uuid()
```

⚙️ **version()**

create version format instance

```
NutsVersionFormat version()
```

⚙️ **xml()**

create xml format instance

```
NutsXmlFormat xml()
```

Command Line

☕ NutsArgument

```
public net.vpc.app.nuts.NutsArgument
```

Command Line Argument

ⓘ Instance Properties

📄 ⓘ argumentKey

return new instance (never null) of the key part of the argument. The key does not include neither ! nor // or = argument parts as they are parsed separately. Here example of getArgumentKey result of some arguments

- Argument("key").getArgumentKey() ==> Argument("key")
- Argument("key=value").getArgumentKey() ==> Argument("key")
- Argument("--key=value").getArgumentKey() ==> Argument("--key")
- Argument("--!key=value").getArgumentKey() ==> Argument("--key")
- Argument("--!//key=value").getArgumentKey() ==> Argument("--key")

```
[read-only] NutsArgument argumentKey
NutsArgument getArgumentKey()
```

📄 ⓘ argumentOptionName

return option key part excluding prefix ('-' and '--')

```
[read-only] NutsArgument argumentOptionName
NutsArgument getArgumentOptionName()
```

📄 ⓘ argumentValue

return new instance (never null) of the value part of the argument (after =). However Argument's

value may be null (` `getArgumentValue().getString() == null` `). Here are some examples of getArgumentValue() result for some common arguments

- Argument("key").getArgumentValue() ==> Argument(null)
- Argument("key=value").getArgumentValue() ==> Argument("value")
- Argument("key=").getArgumentValue() ==> Argument("")
- Argument("--key=value").getArgumentValue() ==> Argument("value")
- Argument("--!key=value").getArgumentValue() ==> Argument("value")
- Argument("--!//key=value").getArgumentValue() ==> Argument("value")

```
[read-only] NutsArgument argumentValue  
NutsArgument getArgumentValue()
```

boolean

test if the argument is valid boolean. a valid boolean mush match one of the following regular expressions : "true|enable|enabled|yes|always|y|on|ok|t" : will be evaluated as true boolean. "false|disable|disabled|no|none|never|n|off|ko" : will beevaluated as false boolean. In both cases, this method returns true. Otherwise, it will return false.

```
[read-only] boolean boolean  
boolean isBoolean()
```

booleanValue

parse argument's value as boolean equivalent to `getArgumentValue().getBoolean()`

```
[read-only] boolean booleanValue  
boolean getBooleanValue()
```

double

parse number and return double.

```
[read-only] double double
double getDouble()
```

enabled

false if option is in one of the following forms :

- -//name
- --//name

where name is any valid identifier

```
[read-only] boolean enabled
boolean isEnabled()
```

int

parse number and return integer.

```
[read-only] int int
int getInt()
```

keyValue

true if the argument is in the form key=value

```
[read-only] boolean keyValue
boolean isKeyValue()
```

keyValueSeparator

return query value separator

```
[read-only] String keyValueSeparator
String getKeyValueSeparator()
```

long

parse number and return long.

```
[read-only] long long  
long getLong()
```

negated

true if option is in one of the following forms :

- `-!name[=...[`
- `--!name[=...[`
- `!name[=...[`

where name is any valid identifier

```
[read-only] boolean negated  
boolean isNegated()
```

nonOption

true if the argument do not start with '-' or '+' or is blank. this is equivalent to ` !isOption()` .

```
[read-only] boolean nonOption  
boolean isNonOption()
```

option

true if the argument starts with '-' or '+'

```
[read-only] boolean option  
boolean isOption()
```

string

string representation of the argument or null

```
[read-only] String string
String getString()
```

stringKey

return key part (never null) of the argument. The key does not include neither ! nor // or = argument parts as they are parsed separately. Here are some examples of getStringKey() result for some common arguments

- Argument("key").getArgumentKey() ==> "key"
- Argument("key=value").getArgumentKey() ==> "key"
- Argument("--key=value").getArgumentKey() ==> "--key"
- Argument("--!key=value").getArgumentKey() ==> "--key"
- Argument("--!//key=value").getArgumentKey() ==> "--key"
- Argument("--//!key=value").getArgumentKey() ==> "--key"

equivalent to `getArgumentKey().getString()`

```
[read-only] String stringKey
String getStringKey()
```

stringOptionName

return option key part excluding prefix ('-' and '--')

```
[read-only] String stringOptionName
String getStringOptionName()
```

stringOptionPrefix

return option prefix part ('-' and '--')

```
[read-only] String stringOptionPrefix  
String getStringOptionPrefix()
```

📄 stringValue

equivalent to `getArgumentValue().getString()`

```
[read-only] String stringValue  
String getStringValue()
```

⚙️ Instance Methods

⚙️ getBoolean(defaultValue)

return boolean value if the current argument can be parsed as valid boolean of defaultValue if not

"true|enable|enabled|yes|always|yon|ok|t" are considered
'true'."false|disable|disabled|no|none|never|n|off|ko" are considered 'false'.

```
Boolean getBoolean(Boolean defaultValue)
```

defaultValue

⚙️ getDouble(defaultValue)

parse number and return double or `defaultValue` if not parsable.

```
double getDouble(double defaultValue)
```

defaultValue

⚙️ getInt(defaultValue)

parse number and return integer or `defaultValue` if not parsable.

```
int getInt(int defaultValue)
```

defaultValue

⚙️ `getLong(defaultValue)`

parse number and return long or `defaultValue` if not parsable.

```
long getLong(long defaultValue)
```

defaultValue

⚙️ `getString(defaultValue)`

string representation of the argument or the given defaultValue

```
String getString(String defaultValue)
```

returned when this argument references null value

⚙️ `getStringValue(defaultValue)`

equivalent to `getArgumentValue().getString(value)`

```
String getStringValue(String defaultValue)
```

default value

⚙️ `required()`

Throw an exception if the argument is null

```
NutsArgument required()
```

☕ NutsArgumentCandidate

```
public net.vpc.app.nuts.NutsArgumentCandidate
```

Argument Candidate used in Auto Complete.

Created by vpc on 3/7/17.

⌚ Instance Properties

📋 📄 display

human display

```
[read-only] String display  
String getDisplay()
```

📋 📄 value

argument value

```
[read-only] String value  
String getValue()
```

☕ NutsArgumentName

```
public net.vpc.app.nuts.NutsArgumentName
```

Non Option Argument specification

⌚ Instance Properties

📋 📄 name

argument name

```
[read-only] String name
String getName()
```

⚙ Instance Methods

⚙ getCandidates(context)

argument candidate values

```
List<NutsArgumentCandidate> get Candidates(NutsCommandAutoComplete context)
```

autocomplete

🖨 NutsCommandAutoCompleteBase

```
public abstract net.vpc.app.nuts.NutsCommandAutoCompleteBase
```

Base (Abstract) implementation of NutsCommandAutoComplete

⌚ Instance Properties

⌚ candidates

candidates map

```
[read-only] public LinkedHashMap<String,NutsArgumentCandidate> candidates
private final LinkedHashMap<String,NutsArgumentCandidate> candidates = new
LinkedHashMap<>()
public List<NutsArgumentCandidate> get Candidates()
```

⌚ workspace

```
[read-only] public NutsWorkspace workspace
public NutsWorkspace get Workspace()
```

⚙ Instance Methods

⚙ addCandidate(value)

add candidate

```
void addCandidate(NutsArgumentCandidate value)
```

candidate

⚙ addCandidatesImpl(value)

simple add candidates implementation

```
NutsArgumentCandidate addCandidatesImpl(NutsArgumentCandidate value)
```

candidate

⚙ get(t)

```
T get(Class<T> t)
```

null

☕ NutsCommandExecOptions

```
public net.vpc.app.nuts.NutsCommandExecOptions
```

Command execution options

CONSTANT FIELDS

serialVersionUID

```
private static final long serialVersionUID = 1
```

Instance Properties

directory

execution directory

```
[read-write] String public directory  
private String directory  
public String getDirectory()  
public NutsCommandExecOptions setDirectory(directory)
```

env

execution environment variables

```
[read-write] Map<String, String> public env  
private Map<String, String> env  
public Map<String, String> getEnv()  
public NutsCommandExecOptions setEnv(env)
```

executionType

execution type

```
[read-write] NutsExecutionType public executionType  
private NutsExecutionType executionType  
public NutsExecutionType getExecutionType()  
public NutsCommandExecOptions setExecutionType(executionType)
```

executorOptions

execution options

```
[read-write] String[] public executorOptions  
private String[] executorOptions  
public String[] getExecutorOptions()  
public NutsCommandExecOptions setExecutorOptions(executorOptions)
```

failFast

when fail fast, non zero exit value will raise NutsExecutionException

```
[read-write] boolean public failFast  
private boolean failFast  
public boolean isFailFast()  
public NutsCommandExecOptions setFailFast(failFast)
```

Instance Methods

equals(o)

```
boolean equals(Object o)
```

null

hashCode()

```
int hashCode()
```

toString()

```
String toString()
```

NutsCommandLine

```
public net.vpc.app.nuts.NutsCommandLine
```

Simple Command line parser implementation. The command line supports arguments in the following forms :

- non option arguments : any argument that does not start with '-'

* long option arguments : any argument that starts with a single '--' in the form of

```
--[//][!]?[^=]*[=.*]
```

- // means disabling the option
- ! means switching (to 'false') the option's value
- the string before the '=' is the option's key
- the string after the '=' is the option's value

Examples :

- --!enable : option 'enable' with 'false' value
- --enable=yes : option 'enable' with 'yes' value
- --!enable=yes : invalid option (no error will be thrown but the result is undefined)

* simple option arguments : any argument that starts with a single '-' in the form of

```
-[//][!]?[a-z][=.*]
```

- -!enable (with expandSimpleOptions=false) : option 'enable' with 'false' value
- --enable=yes : option 'enable' with 'yes' value
- --!enable=yes : invalid option (no error will be thrown but the result is undefined)

* condensed simple option arguments : any argument that starts with a single '-' in the form of

```
-[//]([!]?[a-z])+[=.*]
```

options and is parsable when expandSimpleOptions=true. When activating expandSimpleOptions, multi characters key will be expanded as multiple separate simple options Examples :

- -!enable (with expandSimpleOptions=false) : option 'enable' with 'false' value
- --enable=yes : option 'enable' with 'yes' value
- --!enable=yes : invalid option (no error will be thrown but the result is undefined)
- long option arguments : any argument that starts with a '--'

option may start with '!' to switch armed flags expandSimpleOptions : when activated

Instance Properties

arguments

reset this instance with the given arguments

```
[write-only] NutsCommandLine arguments  
NutsCommandLine setArguments(arguments)
```

autoComplete

set autocomplete instance

```
[read-write] NutsCommandLine autoComplete  
NutsCommandAutoComplete getAutoComplete()  
NutsCommandLine setAutoComplete(autoComplete)
```

autoCompleteMode

true if auto complete instance is registered (is not null)

```
[read-only] boolean autoCompleteMode  
boolean isAutoCompleteMode()
```

commandName

set command name that will be used as an extra info in thrown exceptions

```
[read-write] NutsCommandLine commandName  
String getCommandName()  
NutsCommandLine setCommandName(commandName)
```

empty

true if no more arguments are available

```
[read-only] boolean empty  
boolean isEmpty()
```

execMode

true if auto complete instance is not registered (is null)

```
[read-only] boolean execMode  
boolean isExecMode()
```

expandSimpleOptions

enable or disable simple option expansion

```
[read-write] NutsCommandLine expandSimpleOptions  
boolean isExpandSimpleOptions()  
NutsCommandLine setExpandSimpleOptions(expand)
```

specialSimpleOptions

list of registered simple options

```
[read-only] String[] specialSimpleOptions  
String[] getSpecialSimpleOptions()
```

wordIndex

current word index

```
[read-only] int wordIndex  
int getWordIndex()
```

Instance Methods

⚙️ **accept(values)**

true if arguments start with the given suite.

```
boolean accept(String values)
```

arguments suite

⚙️ **accept(index, values)**

true if arguments start at index `index` with the given suite.

```
boolean accept(int index, String values)
```

starting index arguments suite

⚙️ **contains(name)**

return true if any argument is equal to the given name

```
boolean contains(String name)
```

argument name

⚙️ **find(name)**

find first argument with argument key name

```
NutsArgument find(String name)
```

argument key name

⚙️ **get(index)**

return argument at given index

```
NutsArgument get(int index)
```

argument index

⚙ **hasNext()**

true if there still at least one argument to consume

```
boolean hasNext()
```

⚙ **indexOf(name)**

first argument index (or -1 if not found) with value `name`

```
int indexOf(String name)
```

argument key name

⚙ **isNonOption(index)**

true if the argument and index exists and is non option

```
boolean isNonOption(int index)
```

index

⚙ **isOption(index)**

true if the argument and index exists and is option

```
boolean isOption(int index)
```

index

⚙️ `isSpecialSimpleOption(option)`

test if the option is a registered simple option This method helps considering '-version' as a single simple options when `isExpandSimpleOptions()==true`

```
boolean isSpecialSimpleOption(String option)
```

option

⚙️ `length()`

number of arguments available to retrieve

```
int length()
```

⚙️ `next()`

consume (remove) the first argument and return it return null if not argument is left

```
NutsArgument next()
```

⚙️ `next(name)`

consume (remove) the first argument and return it while adding a hint to Auto Complete about expected argument candidates return null if not argument is left

```
NutsArgument next(NutsArgumentName name)
```

expected argument name

⚙️ `next(names)`

next argument with any value type (may having not a value). equivalent to `next(NutsArgumentType.ANY,names)`

```
NutsArgument next(String names)
```

names

 **next(expectValue, names)**

next argument with any value type (may having not a value).

```
NutsArgument next(NutsArgumentType expectValue, String names)
```

expected value type names

 **nextBoolean(names)**

next argument with boolean value equivalent to next(NutsArgumentType.STRING,names)

```
NutsArgument nextBoolean(String names)
```

names

 **nextNonOption()**

next argument if it exists and it is a non option. Return null in all other cases.

```
NutsArgument nextNonOption()
```

 **nextNonOption(name)**

next argument if it exists and it is a non option. Return null in all other cases.

```
NutsArgument nextNonOption(NutsArgumentName name)
```

argument specification (may be null)

⚙️ `nextRequiredNonOption(name)`

next argument if it exists and it is a non option. Throw an error in all other cases.

```
NutsArgument nextRequiredNonOption(NutsArgumentName name)
```

argument specification (may be null)

⚙️ `nextString(names)`

next argument with string value. equivalent to next(NutsArgumentType.STRING,names)

```
NutsArgument nextString(String names)
```

names

⚙️ `parseLine(commandLine)`

reset this instance with the given parsed arguments

```
NutsCommandLine parseLine(String commandLine)
```

to parse

⚙️ `peek()`

the first argument to consume without removing/consuming it or null if no argument is left

```
NutsArgument peek()
```

⚙️ `process(defaultConfigurable, processor)`

run the processor and fall back to defaultConfigurable

```
void process(NutsConfigurable defaultConfigurable, NutsCommandLineProcessor  
processor)
```

default configurable processor

⚙️ **pushBack(arg)**

push back argument so that it will be first to be retrieved (using next methods)

NutsCommandLine **pushBack(NutsArgument arg)**

argument

⚙️ **registerSpecialSimpleOption(option)**

register `options` as simple (with simple '-') option. This method helps considering '-version' as a single simple options when `isExpandSimpleOptions()==true`

NutsCommandLine **registerSpecialSimpleOption(String option)**

option

⚙️ **requireNonOption()**

throw exception if command line is empty or the first word is an option

NutsCommandLine **requireNonOption()**

⚙️ **required()**

throw exception if command line is empty

NutsCommandLine **required()**

⚙️ **required(errorMessage)**

throw exception if command line is empty

```
NutsCommandLine required(String errorMessage)
```

message to throw

⚙ skip()

skip next argument

```
int skip()
```

⚙ skip(count)

consume `count` words and return how much it was able to consume

```
int skip(int count)
```

count

⚙ skipAll()

consume all words and return consumed count

```
int skipAll()
```

⚙ toArray()

returns un-parsed (or partially parsed) available arguments

```
String[] toArray()
```

⚙ unexpectedArgument()

throw exception if command line is not empty

```
NutsCommandLine unexpectedArgument()
```

⚙️ `unexpectedArgument(errorMessage)`

throw exception if command line is not empty

```
NutsCommandLine unexpectedArgument(String errorMessage)
```

message to throw

⚙️ `unregisterSpecialSimpleOption(option)`

unregister `options` as simple (with simple '-') option. This method helps considering '-version' as a single simple options when `isExpandSimpleOptions()==true`

```
NutsCommandLine unregisterSpecialSimpleOption(String option)
```

option

🖨️ `NutsCommandLineFormat`

```
public net.vpc.app.nuts.NutsCommandLineFormat
```

Simple Command line Format

🔗 `Instance Properties`

🔗 `session`

update session

```
[write-only] NutsCommandLineFormat session  
NutsCommandLineFormat setSession(session)
```

value

set command line from parsed string

```
[read-write] NutsCommandLineFormat value  
NutsCommandLine getValue()  
NutsCommandLineFormat setValue(args)
```

Instance Methods

configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsCommandLineFormat configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

create(args)

return new Command line instance

```
NutsCommandLine create(String args)
```

command line args

create(args)

return new Command line instance

```
NutsCommandLine create(List<String> args)
```

command line args

⚙️ `createArgument(argument)`

create new argument

`NutsArgument createArgument(String argument)`

new argument

⚙️ `createCandidate(value)`

create argument candidate

`NutsArgumentCandidate createCandidate(String value)`

candidate value

⚙️ `createCandidate(value, label)`

create argument candidate

`NutsArgumentCandidate createCandidate(String value, String label)`

candidate value candidate label

⚙️ `createName(type)`

create argument name

`NutsArgumentName createName(String type)`

create argument type

⚙️ `createName(type, label)`

create argument name

```
NutsArgumentName createName(String type, String label)
```

argument type argument label

⚙ **parse**(line)

return new Command line instance

```
NutsCommandLine parse(String line)
```

command line to parse

↳ **NutsCommandLineProcessor**

```
public net.vpc.app.nuts.NutsCommandLineProcessor
```

The processor is called to process the command line arguments.

- `init` : called initially
- `processOption` | `processNonOption` : called multiple times until the command line is consumed
- `prepare` : called when the command line is fully consumed
- `exec` | `autoComplete` : called to process execution of autocomplete

⚙ **Instance Methods**

⚙ **autoComplete(autoComplete)**

called when auto-complete (`autoComplete` is not null)

```
void autoComplete(NutsCommandAutoComplete autoComplete)
```

autoComplete instance

⚙ exec()

execute options, called after all options was processed and cmdLine.isExecMode() return true.

```
void exec()
```

⚙ init(commandline)

initialize the processor

```
void init(NutsCommandLine commandline)
```

associated commandline

⚙ nextNonOption(nonOption, cmdLine)

process the given non option argument that was peeked from the command line. Implementations MUST call one of the "next" methods to

```
boolean nextNonOption(NutsArgument nonOption, NutsCommandLine cmdLine)
```

peeked argument associated commandline

⚙ nextOption(option, cmdLine)

process the given option argument that was peeked from the command line. Implementations MUST call one of the "next" methods to

```
boolean nextOption(NutsArgument option, NutsCommandLine cmdLine)
```

peeked argument associated commandline

⚙ prepare(commandline)

prepare for execution or for auto-complete

```
void prepare(NutsCommandLine commandline)
```

associated commandline

⌚ NutsConfigurable

```
public net.vpc.app.nuts.NutsConfigurable
```

Configurable interface define a extensible way to configure nuts commands and objects using simple argument line options.

⚙ Instance Methods

⚙ configure(skipUnsupported, args)

configure the current command with the given arguments.

```
Object configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped silently arguments to configure with

⚙ configure(skipUnsupported, commandLine)

configure the current command with the given arguments.

```
boolean configure(boolean skipUnsupported, NutsCommandLine commandLine)
```

when true, all unsupported options are skipped silently arguments to configure with

⚙ configureFirst(commandLine)

ask `this` instance to configure with the very first argument of `commandLine`. If the first argument is not supported, return `false` and consume (skip/read) the argument. If the argument required one or more parameters, these arguments are also consumed and finally return `true`

```
boolean configureFirst(NutsCommandLine commandLine)
```

arguments to configure with

Commands

☕ NutsDeployCommand

```
public net.vpc.app.nuts.NutsDeployCommand
```

Nuts deploy command

Instance Properties

content

set content

```
[write-only] NutsDeployCommand content  
NutsDeployCommand setContent(url)
```

descSha1

set descriptor sha1 hash

```
[write-only] NutsDeployCommand descSha1  
NutsDeployCommand setDescSha1(descSHA1)
```

descriptor

set descriptor

```
[write-only] NutsDeployCommand descriptor  
NutsDeployCommand setDescriptor(descriptor)
```

📄 ids

return ids to deploy from source repository

```
[read-only] NutsId[] ids  
NutsId[] getIds()
```

repository

set target repository to deploy to

```
[write-only] NutsDeployCommand repository  
NutsDeployCommand setRepository(repository)
```

result

run command (if not yet run) and return result

```
[read-only] NutsId[] result  
NutsId[] getResult()
```

session

update session

```
[write-only] NutsDeployCommand session  
NutsDeployCommand setSession(session)
```

sha1

set content sha1 hash

```
[read-write] NutsDeployCommand sha1  
String getSha1()  
NutsDeployCommand setSha1(sh1)
```

sourceRepository

set source repository to deploy from the given ids

```
[write-only] NutsDeployCommand sourceRepository  
NutsDeployCommand setSourceRepository(repository)
```

targetRepository

set target repository to deploy to

```
[read-write] NutsDeployCommand targetRepository  
String getTargetRepository()  
NutsDeployCommand setTargetRepository(repository)
```

Instance Methods

addId(id)

add id to deploy from source repository

```
NutsDeployCommand addId(String id)
```

id to deploy from source repository

addId(id)

add id to deploy from source repository

```
NutsDeployCommand addId(NutsId id)
```

id to deploy from source repository

addIds(values)

add ids to deploy from source repository

```
NutsDeployCommand addIds(NutsId values)
```

ids to deploy from source repository

⚙️ addIds(values)

add ids to deploy from source repository

```
NutsDeployCommand addIds(String values)
```

ids to deploy from source repository

⚙️ clearIds()

reset ids list to deploy

```
NutsDeployCommand clearIds()
```

⚙️ configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsDeployCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ copySession()

copy session

```
NutsDeployCommand copySession()
```

⚙️ from(repository)

set source repository to deploy from the given ids

```
NutsDeployCommand from(String repository)
```

source repository to deploy from

⚙️ `removeld(id)`

remove id to deploy from source repository

```
NutsDeployCommand removeId(String id)
```

id to undo deploy from source repository

⚙️ `removeld(id)`

remove id to deploy from source repository

```
NutsDeployCommand removeId(NutsId id)
```

id to undo deploy from source repository

⚙️ `run()`

execute the command and return this instance

```
NutsDeployCommand run()
```

⚙️ `to(repository)`

set target repository to deploy to

```
NutsDeployCommand to(String repository)
```

target repository to deploy to

☕️ `NutsExecCommand`

```
public net.vpc.app.nuts.NutsExecCommand
```

Execute command. This class helps executing all types of executables : internal, external, alias and system

Instance Properties

command

set command artifact definition. The definition is expected to include content, dependencies, effective descriptor and install information.

```
[read-write] NutsExecCommand command
String[] getCommand()
NutsExecCommand setCommand(definition)
```

directory

set execution directory

```
[read-write] NutsExecCommand directory
String getDirectory()
NutsExecCommand setDirectory(directory)
```

dry

if true set dry execution

```
[read-write] NutsExecCommand dry
boolean isDry()
NutsExecCommand setDry(value)
```

env

clear existing env and set new env

```
[read-write] NutsExecCommand env
Map<String, String> getEnv()
NutsExecCommand setEnv(env)
```

err

set new command error stream (standard error destination)

```
[read-write] NutsExecCommand err  
PrintStream getErr()  
NutsExecCommand setErr(err)
```

errorString

return grabbed error after command execution

```
[read-only] String errorString  
String getErrorString()
```

executionType

set execution type

```
[read-write] NutsExecCommand executionType  
NutsExecutionType getExecutionType()  
NutsExecCommand setExecutionType(executionType)
```

executorOptions

return executor options

```
[read-only] String[] executorOptions  
String[] getExecutorOptions()
```

failFast

when the execution returns a non zero result, an exception is thrown. Particularly, if grabOutputString is used, error exception will state the output message

```
[read-write] NutsExecCommand failFast  
boolean isFailFast()  
NutsExecCommand setFailFast(failFast)
```



set new command input stream (standard input source)

```
[read-write] NutsExecCommand in  
InputStream getIn()  
NutsExecCommand setIn(in)
```



set new command output stream (standard output destination)

```
[read-write] NutsExecCommand out  
PrintStream getOut()  
NutsExecCommand setOut(out)
```



return grabbed output after command execution

```
[read-only] String outputString  
String getOutputString()
```



if true redirect standard error is redirected to standard output

```
[read-write] NutsExecCommand redirectErrorStream  
boolean isRedirectErrorStream()  
NutsExecCommand setRedirectErrorStream(redirectErrorStream)
```

result

return result value. if not yet executed, will execute first.

```
[read-only] int result  
int getResult()
```

resultException

return result exception or null

```
[read-only] NutsExecutionException resultException  
NutsExecutionException getResultException()
```

session

update session

```
[write-only] NutsExecCommand session  
NutsExecCommand setSession(session)
```

Instance Methods

addCommand(command)

append command arguments

```
NutsExecCommand addCommand(String command)
```

command

addCommand(command)

append command arguments

```
NutsExecCommand addCommand(Collection<String> command)
```

command

⚙ `addEnv(env)`

merge env properties

```
NutsExecCommand addEnv(Map<String, String> env)
```

env properties

⚙ `addExecutorOption(executorOption)`

append executor options

```
NutsExecCommand addExecutorOption(String executorOption)
```

executor options

⚙ `addExecutorOptions(executorOptions)`

append executor options

```
NutsExecCommand addExecutorOptions(String executorOptions)
```

executor options

⚙ `addExecutorOptions(executorOptions)`

append executor options

```
NutsExecCommand addExecutorOptions(Collection<String> executorOptions)
```

executor options

⚙ `clearCommand()`

clear command

NutsExecCommand **clearCommand()**

⚙️ **clearEnv()**

clear env

NutsExecCommand **clearEnv()**

⚙️ **clearExecutorOptions()**

clear executor options

NutsExecCommand **clearExecutorOptions()**

⚙️ **configure(skipUnsupported, args)**

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

NutsExecCommand **configure(boolean skipUnsupported, String args)**

when true, all unsupported options are skipped argument to configure with

⚙️ **copy()**

create a copy of `this` instance

NutsExecCommand **copy()**

⚙️ **copyFrom(other)**

copy all field from the given command into `this` instance

NutsExecCommand **copyFrom(NutsExecCommand other)**

command to copy from

⚙ `copySession()`

copy session

NutsExecCommand `copySession()`

⚙ `embedded()`

set embedded execution type

NutsExecCommand `embedded()`

⚙ `format()`

NutsExecCommandFormat `format()`

⚙ `grabErrorString()`

grab to memory standard error

NutsExecCommand `grabErrorString()`

⚙ `grabOutputString()`

grab to memory standard output

NutsExecCommand `grabOutputString()`

⚙ `rootCmd()`

set root command execution type

```
NutsExecCommand rootCmd()
```

⚙ **run()**

execute the command and return this instance

```
NutsExecCommand run()
```

⚙ **setEnv(key, value)**

set or unset env property. the property is unset if the value is null.

```
NutsExecCommand setEnv(String key, String value)
```

env key env value

⚙ **spawn()**

set spawn execution type

```
NutsExecCommand spawn()
```

⚙ **userCmd()**

set user command execution type

```
NutsExecCommand userCmd()
```

⚙ **which()**

return executable information

```
NutsExecutableInformation which()
```

 NutsFetchCommand

```
public net.vpc.app.nuts.NutsFetchCommand
```

Fetch command class helps fetching/retrieving a artifact with all of its files.

 Instance Properties  cached

enable/disable retrieval from cache

```
[read-write] NutsFetchCommand cached
boolean isCached()
NutsFetchCommand setCached(enable)
```

  content

enable/disable retrieval of content info

```
[read-write] NutsFetchCommand content
boolean isContent()
NutsFetchCommand setContent(enable)
```

  dependencies

enable/disable dependencies list retrieval

```
[read-write] NutsFetchCommand dependencies
boolean isDependencies()
NutsFetchCommand setDependencies(enable)
```

  dependenciesTree

enable/disable dependencies tree retrieval

```
[read-write] NutsFetchCommand dependenciesTree  
boolean isDependenciesTree()  
NutsFetchCommand setDependenciesTree(enable)
```

effective

enable/disable effective descriptor evaluation

```
[read-write] NutsFetchCommand effective  
boolean isEffective()  
NutsFetchCommand setEffective(enable)
```

failFast

set armed (or disarmed) fail safe mode. if true, null replaces NutsNotFoundException.

```
[read-write] NutsFetchCommand failFast  
boolean isFailFast()  
NutsFetchCommand setFailFast(enable)
```

fetchStrategy

set fetch strategy.

```
[read-write] NutsFetchCommand fetchStrategy  
NutsFetchStrategy getFetchStrategy()  
NutsFetchCommand setFetchStrategy(fetchStrategy)
```

id

set id to fetch.

```
[read-write] NutsFetchCommand id  
NutsId getId()  
NutsFetchCommand setId(id)
```

indexed

set index filter. if null index is removed. if false do not consider index. if true, consider index.

```
[read-write] NutsFetchCommand indexed  
boolean isIndexed()  
NutsFetchCommand setIndexed(enable)
```

installed

search for installed/non installed packages

```
[read-write] NutsFetchCommand installed  
Boolean getInstalled()  
NutsFetchCommand setInstalled(value)
```

location

set locating where to fetch the artifact. If the location is a folder, a new name will be generated.

```
[read-write] NutsFetchCommand location  
Path getLocation()  
NutsFetchCommand setLocation(fileOrFolder)
```

optional

set option filter. if null filter is removed. if false only non optional will be retrieved. if true, only optional will be retrieved.

```
[read-write] NutsFetchCommand optional  
Boolean getOptional()  
NutsFetchCommand setOptional(enable)
```

resultContent

return result as content

```
[read-only] NutsContent resultContent  
NutsContent getResultContent()
```

resultContentHash

return result as content hash string

```
[read-only] String resultContentHash  
String getResultContentHash()
```

resultDefinition

return result as artifact definition

```
[read-only] NutsDefinition resultDefinition  
NutsDefinition getResultDefinition()
```

resultDescriptor

return result as descriptor

```
[read-only] NutsDescriptor resultDescriptor  
NutsDescriptor getResultDescriptor()
```

resultDescriptorHash

return result as descriptor hash string

```
[read-only] String resultDescriptorHash  
String getResultDescriptorHash()
```

resultId

return result as id

```
[read-only] NutsId resultId  
NutsId getResultSetId()
```

📄📄 resultPath

return result as content path

```
[read-only] Path resultPath  
Path getResultSetPath()
```

📄📄 scope

dependencies scope filters

```
[read-only] Set<NutsDependencyScope> scope  
Set<NutsDependencyScope> getScope()
```

@update session

```
[write-only] NutsFetchCommand session  
NutsFetchCommand setSession(session)
```

📝📝 transitive

set or unset transitive mode

```
[read-write] NutsFetchCommand transitive  
boolean isTransitive()  
NutsFetchCommand setTransitive(enable)
```

⚙️ Instance Methods

addRepositories(value)

add repository filter

```
NutsFetchCommand addRepositories(Collection<String> value)
```

repository filter

addRepositories(values)

add repository filter

```
NutsFetchCommand addRepositories(String values)
```

repository filter

addRepository(value)

add repository filter

```
NutsFetchCommand addRepository(String value)
```

repository filter

addScope(scope)

add dependency scope filter. Only relevant with `#setDependencies(boolean)` `` and `#setDependenciesTree(boolean)` ``

```
NutsFetchCommand addScope(NutsDependencyScopePattern scope)
```

scope filter

addScope(scope)

add dependency scope filter. Only relevant with `#setDependencies(boolean)` `` and `#setDependenciesTree(boolean)` ``

NutsFetchCommand **addScope**(NutsDependencyScope scope)

scope filter

⚙️ **addScopes(scope)**

add dependency scope filter. Only relevant with `#setDependencies(boolean)` and `#setDependenciesTree(boolean)`

NutsFetchCommand **addScopes**(NutsDependencyScope scope)

scope filter

⚙️ **addScopes(scope)**

add dependency scope filter. Only relevant with `#setDependencies(boolean)` and `#setDependenciesTree(boolean)`

NutsFetchCommand **addScopes**(NutsDependencyScopePattern scope)

scope filter

⚙️ **clearRepositories()**

remove all repository filters

NutsFetchCommand **clearRepositories()**

⚙️ **clearScopes()**

remove all dependency scope filters.

NutsFetchCommand **clearScopes()**

`⚙️ configure(skipUnsupported, args)`

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)`

to help return a more specific return type;

`NutsFetchCommand configure(boolean skipUnsupported, String args)`

when true, all unsupported options are skipped
argument to configure with

⚙️ `copy()`
create copy (new instance) of ```this``` command

`NutsFetchCommand copy()`

⚙️ `copyFrom(other)`
copy into ```this```` from ```other```` fetch command

`NutsFetchCommand copyFrom(NutsFetchCommand other)`

copy into {@code this} from {@code other} fetch command

⚙️ `copySession()`
copy session

`NutsFetchCommand copySession()`

⚙️ `installed()`
search for non installed packages

`NutsFetchCommand installed()`

⚙️ `installed(value)`
search for installed/non installed packages

NutsFetchCommand installed(Boolean value)

```
new value  
  
#### ⚙️ notInstalled()  
search for non installed packages
```

NutsFetchCommand notInstalled()

```
#### ⚙️ removeRepository(value)  
remove repository filter
```

NutsFetchCommand removeRepository(String value)

```
repository filter  
  
#### ⚙️ removeScope(scope)  
remove dependency scope filter.
```

NutsFetchCommand removeScope(NutsDependencyScope scope)

```
scope filter  
  
#### ⚙️ removeScope(scope)  
remove dependency scope filter.
```

NutsFetchCommand removeScope(NutsDependencyScopePattern scope)

```
scope filter  
  
#### ⚙️ run()  
execute the command and return this instance
```

NutsFetchCommand run()

```
#### ⚙️ setAnyWhere()  
all artifacts (local and remote). If local result found will any way  
fetch remote.
```

NutsFetchCommand setAnyWhere()

```
#### ⚙️ setDefaultLocation()  
unset location to store to fetched id and fall back to default location.
```

NutsFetchCommand setDefaultLocation()

```
#### ⚙️ setNutsApi()  
set id to fetch to nuts-api (api artifact)
```

NutsFetchCommand setNutsApi()

```
#### ⚙️ setNutsRuntime()  
set id to fetch to nuts-core (runtime artifact)
```

NutsFetchCommand setNutsRuntime()

```
#### ⚙️ setOffline()  
local only (installed or not)
```

NutsFetchCommand setOffline()

```
#### ⚙️ setOnline()  
local or remote. If local result found will not fetch remote.
```

NutsFetchCommand setOnline()

```
#### ⚙️ setRemote()  
remote only
```

NutsFetchCommand setRemote()

```
## ☕ NutsPushCommand
```

```
public net.vpc.app.nuts.NutsPushCommand
```

Push command
☰ Instance Properties
📋 ☰ args
 return all arguments to pass to the push command

[read-only[String[] args String[] getArgs()

📋 ☰ ids
 return ids to push for

[read-only[NutsId[] ids NutsId[] getIds()

📋 ☰ lockedIds
 return locked ids to prevent them to be updated or the force other ids to use them (the installed version).

[read-only[NutsId[] lockedIds NutsId[] getLockedIds()

📋 ☰ offline
 local only (installed or not)

[read-write[NutsPushCommand offline boolean isOffline() NutsPushCommand setOffline(boolean)

📋 ☰ repository
 repository to push from

[read-write[NutsPushCommand repository String getRepository() NutsPushCommand setRepository(repository)

☰ session
 update session

[write-only[NutsPushCommand session NutsPushCommand setSession(session)

```
### ⚙ Instance Methods  
#### ⚙ addArg(arg)  
add argument to pass to the push command
```

NutsPushCommand addArg(String arg)

```
argument  
  
#### ⚙ addArgs(args)  
add arguments to pass to the push command
```

NutsPushCommand addArgs(String args)

```
argument  
  
#### ⚙ addArgs(args)  
add arguments to pass to the push command
```

NutsPushCommand addArgs(Collection<String> args)

```
argument  
  
#### ⚙ addId(id)  
add id to push.
```

NutsPushCommand addId(NutsId id)

```
id to push  
  
#### ⚙ addId(id)  
add id to push.
```

NutsPushCommand addId(String id)

id to push

```
#### ⚙ addIds(ids)  
add ids to push.
```

NutsPushCommand addIds(NutsId ids)

id to push

```
#### ⚙ addIds(ids)  
add ids to push.
```

NutsPushCommand addIds(String ids)

id to push

```
#### ⚙ addLockedId(id)  
add locked ids to prevent them to be updated or the force other ids to use them  
(the installed version).
```

NutsPushCommand addLockedId(NutsId id)

id to lock

```
#### ⚙ addLockedId(id)  
add locked ids to prevent them to be updated or the force other ids to use them  
(the installed version).
```

NutsPushCommand addLockedId(String id)

id to lock

```
#### ⚙ addLockedIds(values)  
add locked ids to prevent them to be updated or the force other ids to use them  
(the installed version).
```

NutsPushCommand addLockedIds(NutsId values)

id to lock

```
#### ⚙ addLockedIds(values)  
define locked ids to prevent them to be updated or the force other ids to use  
them (the installed version).
```

NutsPushCommand addLockedIds(String values)

ids

```
#### ⚙ arg(arg)  
add argument to pass to the push command
```

NutsPushCommand arg(String arg)

argument

```
#### ⚙ args(args)  
add arguments to pass to the push command
```

NutsPushCommand args(String args)

argument

```
#### ⚙ args(args)  
add arguments to pass to the push command
```

NutsPushCommand args(Collection<String> args)

argument

```
#### ⚙ clearArgs()  
clear all arguments to pass to the push command
```

NutsPushCommand clearArgs()

```
#### ⚙ clearIds()  
reset ids to push for
```

NutsPushCommand clearIds()

```
#### ⚙ clearLockedIds()  
reset locked ids
```

NutsPushCommand clearLockedIds()

```
#### ⚙ configure(skipUnsupported, args)  
configure the current command with the given arguments. This is an  
override of the ``` NutsConfigurable#configure(boolean, java.lang.String...) ````  
to help return a more specific return type;
```

NutsPushCommand configure(boolean skipUnsupported, String args)

when true, all unsupported options are skipped
argument to configure with

```
#### ⚙ copySession()  
copy session
```

NutsPushCommand copySession()

```
#### ⚙ id(id)  
add id to push.
```

NutsPushCommand id(NutsId id)

id to push

```
#### ⚙ id(id)  
add id to push.
```

NutsPushCommand id(String id)

id to push

```
#### ⚙ ids(ids)  
add ids to push.
```

NutsPushCommand ids(NutsId ids)

id to push

```
#### ⚙ ids(ids)  
add ids to push.
```

NutsPushCommand ids(String ids)

id to push

```
#### ⚙ lockedId(id)  
add locked ids to prevent them to be updated or the force other ids to use them  
(the installed version).
```

NutsPushCommand lockedId(NutsId id)

id to lock

```
#### ⚙ lockedId(id)  
add locked ids to prevent them to be updated or the force other ids to use them  
(the installed version).
```

NutsPushCommand lockedId(String id)

id to lock

```
#### ⚙ lockedIds(values)  
define locked ids to prevent them to be updated or the force other ids to use  
them (the installed version).
```

NutsPushCommand lockedIds(NutsId values)

ids

```
#### ⚙ lockedIds(values)
define locked ids to prevent them to be updated or the force other ids to use
them (the installed version).
```

NutsPushCommand lockedIds(String values)

ids

```
#### ⚙ offline()
local only (installed or not)
```

NutsPushCommand offline()

```
#### ⚙ offline(boolean offline)
local only (installed or not)
```

NutsPushCommand offline(boolean offline)

enable offline mode

```
#### ⚙ removeId(NutsId id)
remove id to push.
```

NutsPushCommand removeId(NutsId id)

id to push

```
#### ⚙ removeId(String id)
remove id to push.
```

NutsPushCommand removeId(String id)

id to push

⚙ removeLockedId(id)

remove locked ids to prevent them to be updated or the force other ids to use them (the installed version).

NutsPushCommand removeLockedId(NutsId id)

id to unlock

⚙ removeLockedId(id)

remove locked ids to prevent them to be updated or the force other ids to use them (the installed version).

NutsPushCommand removeLockedId(String id)

id to unlock

⚙ repository(repository)

repository to push from

NutsPushCommand repository(String repository)

repository to push from

⚙ run()

execute the command and return this instance

NutsPushCommand run()

☕ NutsRemoveOptions

public net.vpc.app.nuts.NutsRemoveOptions

📄 Instance Properties

✎ 📄 erase

```
[read-write] boolean public eraseprivate boolean erase = false public boolean isErase() public  
NutsRemoveOptions setErase(erase)
```

session

```
[read-write] NutsSession public sessionprivate NutsSession session public NutsSession  
getSession() public NutsRemoveOptions setSession(session)
```

Instance Methods

equals(o)

```
boolean equals(Object o)
```

```
null
```

erase()

```
NutsRemoveOptions erase()
```

erase(erase)

```
NutsRemoveOptions erase(boolean erase)
```

```
null
```

hashCode()

```
int hashCode()
```

toString()

```
String toString()
```

NutsSearchCommand

```
public net.vpc.app.nuts.NutsSearchCommand
```

Search command class helps searching multiple artifacts with all of their files.

 Instance Properties

 application

set nuts app filter. if true nuts app (implementing NutsApplication) only are retrieved.

```
[read-write[ NutsSearchCommand application boolean isApplication() NutsSearchCommand  
setApplication(enable)
```

 arch

```
[read-only[ String[[ arch String[[ getArch()
```

 basePackage

include base package when searching for inlined dependencies

```
[read-write[ NutsSearchCommand basePackage boolean isBasePackage() NutsSearchCommand  
setBasePackage(includeBasePackage)
```

 cached

enable/disable retrieval from cache

```
[read-write[ NutsSearchCommand cached boolean isCached() NutsSearchCommand  
setCached(enable)
```

 companion

set companions filter. if true companions only are retrieved.

```
[read-write[ NutsSearchCommand companion boolean isCompanion() NutsSearchCommand  
setCompanion(enable)
```

 comparator

result comparator

[read-only] Comparator comparator Comparator getComparator()

content
enable/disable retrieval of content info

[read-write] NutsSearchCommand content boolean isContent() NutsSearchCommand
setContent(enable)

defaultVersions
default version only filter

[read-write] NutsSearchCommand defaultVersions Boolean getDefaultVersions()
NutsSearchCommand setDefaultVersions(enable)

dependencies
enable/disable dependencies list retrieval

[read-write] NutsSearchCommand dependencies boolean isDependencies() NutsSearchCommand
setDependencies(enable)

dependenciesTree
enable/disable dependencies tree retrieval

[read-write] NutsSearchCommand dependenciesTree boolean isDependenciesTree()
NutsSearchCommand setDependenciesTree(enable)

dependencyFilter
define dependency filter. applicable when using ```
#setInlineDependencies(boolean)```

[read-write] NutsSearchCommand dependencyFilter NutsDependencyFilter getDependencyFilter()
NutsSearchCommand setDependencyFilter(filter)

descriptorFilter
define descriptor filter.

```
[read-write] NutsSearchCommand descriptorFilter NutsDescriptorFilter getDescriptorFilter()
NutsSearchCommand setDescriptorFilter(filter)
```

distinct
skip duplicates

```
[read-write] NutsSearchCommand distinct boolean isDistinct() NutsSearchCommand
setDistinct(distinct)
```

effective
enable/disable effective descriptor evaluation

```
[read-write] NutsSearchCommand effective boolean isEffective() NutsSearchCommand
setEffective(enable)
```

exec
set app filter. if true non lib (app) only are retrieved.

```
[read-write] NutsSearchCommand exec boolean isExec() NutsSearchCommand setExec(enable)
```

extension
set extensions filter. if true extensions only are retrieved.

```
[read-write] NutsSearchCommand extension boolean isExtension() NutsSearchCommand
setExtension(enable)
```

failFast
set armed (or disarmed) fail safe mode. if true, null replaces
NutsNotFoundException.

```
[read-write] NutsSearchCommand failFast boolean isFailFast() NutsSearchCommand
setFailFast(enable)
```

fetchStrategy
set fetch strategy.

[read-write] NutsSearchCommand fetchStrategy NutsFetchStrategy getFetchStrategy()
 NutsSearchCommand setFetchStrategy(fetchStrategy)

idFilter
 define id filter.

[read-write] NutsSearchCommand idFilter NutsIdFilter getIdFilter() NutsSearchCommand
 setIdFilter(filter)

ids
 return ids to search for

[read-only] NutsId[[ids NutsId[[getIds()

indexed
 set index filter. if null index is removed. if false do not consider
 index. if true, consider index.

[read-write] NutsSearchCommand indexed boolean isIndexed() NutsSearchCommand
 setIndexed(enable)

inlineDependencies
 enable/disable inlined dependencies list retrieval

[read-write] NutsSearchCommand inlineDependencies boolean isInlineDependencies()
 NutsSearchCommand setInlineDependencies(enable)

installStatus
 search for non packages with the given `installStatus`

[read-write] NutsSearchCommand installStatus NutsInstallStatus getInstallStatus()
 NutsSearchCommand setInstallStatus(installStatus)

latest
 if true search must return only latest versions for each artifact id

[read-write[NutsSearchCommand latest boolean isLatest() NutsSearchCommand setLatest(enable)

lib
set lib filter. if true lib (non app) only are retrieved.

[read-write[NutsSearchCommand lib boolean isLib() NutsSearchCommand setLib(enable)

location
set locating where to fetch the artifact. If the location is a folder, a new name will be generated.

[read-write[NutsSearchCommand location Path getLocation() NutsSearchCommand
setLocation(fileOrFolder)

lockedIds
return locked ids to prevent them to be updated or the force other ids to use them (the installed version).

[read-only[NutsId[[lockedIds NutsId[[getLockedIds()

optional
set option filter. if null filter is removed. if false only non optional will be retrieved. if true, only optional will be retrieved.

[read-write[NutsSearchCommand optional Boolean getOptional() NutsSearchCommand
setOptional(enable)

packaging

[read-only[String[[packaging String[[getPackaging()

printResult
enable print search result

[read-write[NutsSearchCommand printResult boolean isPrintResult() NutsSearchCommand
setPrintResult(enable)

repositories

[read-only[String[[repositories String[] getRepositories()

repository
define repository filter.

[write-only[NutsSearchCommand repository NutsSearchCommand setRepository(filter)

repositoryFilter
define repository filter.

[read-write[NutsSearchCommand repositoryFilter NutsRepositoryFilter getRepositoryFilter()
NutsSearchCommand setRepositoryFilter(filter)

resultArchs
return result as archs

[read-only[NutsResultList<String> resultArchs NutsResultList<String> getResultArchs()

resultClassLoader
execute query and return result as class loader

[read-only[ClassLoader resultClassLoader ClassLoader getResultClassLoader()

resultClassPath
execute query and return result as class path string

[read-only[String resultClassPath String getResultClassPath()

resultDefinitions
execute query and return result as definitions

[read-only[NutsResultList<NutsDefinition> resultDefinitions NutsResultList<NutsDefinition> getResultDefinitions()

```
#### 📄 resultExecutionEntries  
return result as execution entries
```

[read-only] NutsResultList<NutsExecutionEntry> resultExecutionEntries
NutsResultList<NutsExecutionEntry> getResultExecutionEntries()

```
#### 📄 resultIds  
execute query and return result as ids
```

[read-only] NutsResultList<NutsId> resultIds NutsResultList<NutsId> getResultIds()

```
#### 📄 resultInstallDates  
execute query and return install dates
```

[read-only] NutsResultList<Instant> resultInstallDates NutsResultList<Instant>
getResultInstallDates()

```
#### 📄 resultInstallFolders  
execute query and return install folders
```

[read-only] NutsResultList<Path> resultInstallFolders NutsResultList<Path>
getResultInstallFolders()

```
#### 📄 resultInstallUsers  
execute query and return install users
```

[read-only] NutsResultList<String> resultInstallUsers NutsResultList<String>
getResultInstallUsers()

```
#### 📄 resultNames  
return result as artifact names
```

[read-only] NutsResultList<String> resultNames NutsResultList<String> getResultNames()

```
#### 📄 resultNutsPath  
execute query and return result as nuts path string
```

[read-only] String resultNutsPath String getResultNutsPath()

```
#### 📄 resultOsdist  
return result as osdist names
```

[read-only] NutsResultList<String> resultOsdist NutsResultList<String> getResultOsdist()

```
#### 📄 resultOses  
return result as operating system names
```

[read-only] NutsResultList<String> resultOses NutsResultList<String> getResultOses()

```
#### 📄 resultPackagings  
return result as packagings
```

[read-only] NutsResultList<String> resultPackagings NutsResultList<String>
getResultPackagings()

```
#### 📄 resultPathNames  
return result as content path names
```

[read-only] NutsResultList<String> resultPathNames NutsResultList<String>
getResultPathNames()

```
#### 📄 resultPaths  
return result as content paths
```

[read-only] NutsResultList<String> resultPaths NutsResultList<String> getResultPaths()

```
#### 📄 resultPlatforms  
return result as platforms
```

```
[read-only] NutsResultList<String> resultPlatforms NutsResultList<String>
getResultPlatforms()
```

```
#### 🖊 runtime  
add runtime id to search
```

```
[read-write] NutsSearchCommand runtime boolean isRuntime() NutsSearchCommand
setRuntime(enable)
```

```
#### 📄 scope  
scope filter filter. applicable with ``` #setInlineDependencies(boolean)```
```

```
[read-only] Set<NutsDependencyScope> scope Set<NutsDependencyScope> getScope()
```

```
#### 📄 scripts  
return javascript filters
```

```
[read-only] String[] scripts String[] getScripts()
```

```
#### 🚧 session  
update session
```

```
[write-only] NutsSearchCommand session NutsSearchCommand setSession(session)
```

```
#### 🖊 sorted  
sort result
```

```
[read-write] NutsSearchCommand sorted boolean isSorted() NutsSearchCommand setSorted(sort)
```

```
#### 🖊 targetApiVersion  
set target api version
```

```
[read-write] NutsSearchCommand targetApiVersion String getTargetApiVersion()
NutsSearchCommand setTargetApiVersion(targetApiVersion)
```

 **transitive**
set or unset transitive mode

[read-write] NutsSearchCommand transitive boolean isTransitive() NutsSearchCommand setTransitive(enable)

 **Instance Methods**
 **addArch(value)**
add arch to search

NutsSearchCommand addArch(String value)

arch to search for

 **addArchs(values)**
add archs to search

NutsSearchCommand addArchs(Collection<String> values)

arch to search for

 **addArchs(values)**
add archs to search

NutsSearchCommand addArchs(String values)

arch to search for

 **addId(id)**
add id to search.

NutsSearchCommand addId(String id)

id to search

 **addId(id)**
add id to search.

NutsSearchCommand addId(NutsId id)

id to search

⚙ addIds(ids)
add ids to search.

NutsSearchCommand addIds(String ids)

id to search

⚙ addIds(ids)
add ids to search.

NutsSearchCommand addIds(NutsId ids)

ids to search

⚙ addLockedId(id)
add locked ids to prevent them to be updated or the force other ids to use them
(the installed version).

NutsSearchCommand addLockedId(NutsId id)

id to lock

⚙ addLockedId(id)
add locked ids to prevent them to be updated or the force other ids to use them
(the installed version).

NutsSearchCommand addLockedId(String id)

id to lock

⚙ addLockedIds(values)
define locked ids to prevent them to be updated or the force other ids to use
them (the installed version).

NutsSearchCommand addLockedIds(String values)

ids

```
#### ⚙ addLockedIds(values)  
define locked ids to prevent them to be updated or the force other ids to use  
them (the installed version).
```

NutsSearchCommand addLockedIds(NutsId values)

ids

```
#### ⚙ addPackaging(value)  
add packaging to search
```

NutsSearchCommand addPackaging(String value)

packaging to search for

```
#### ⚙ addPackagings(values)  
add packagings to search
```

NutsSearchCommand addPackagings(Collection<String> values)

packagings to search for

```
#### ⚙ addPackagings(values)  
add packagings to search
```

NutsSearchCommand addPackagings(String values)

packagings to search for

```
#### ⚙ addRepositories(values)  
add repositories to search into
```

NutsSearchCommand addRepositories(Collection<String> values)

repositories to search into

```
#### ⚙ addRepositories(values)  
add repositories to search into
```

NutsSearchCommand addRepositories(String values)

repositories to search into

```
#### ⚙ addRepository(value)  
add repository to search into
```

NutsSearchCommand addRepository(String value)

repository to search into

```
#### ⚙ addScope(scope)  
add dependency scope filter. Only relevant with ``` #setDependencies(boolean)```  
and ``` #setDependenciesTree(boolean)```
```

NutsSearchCommand addScope(NutsDependencyScope scope)

scope filter

```
#### ⚙ addScope(scope)  
add dependency scope filter. Only relevant with ``` #setDependencies(boolean)```  
and ``` #setDependenciesTree(boolean)```
```

NutsSearchCommand addScope(NutsDependencyScopePattern scope)

scope filter

```
#### ⚙ addScopes(scope)  
add dependency scope filter. Only relevant with ``` #setDependencies(boolean)```  
and ``` #setDependenciesTree(boolean)```
```

NutsSearchCommand addScopes(NutsDependencyScope scope)

scope filter

```
#### ⚙ addScopes(scope)  
add dependency scope filter. Only relevant with ``` #setDependencies(boolean)```  
and ``` #setDependenciesTree(boolean)```
```

NutsSearchCommand addScopes(NutsDependencyScopePattern scope)

scope filter

```
#### ⚙ addScript(value)  
add javascript filter.
```

NutsSearchCommand addScript(String value)

javascript filter

```
#### ⚙ addScripts(value)  
add javascript filter.
```

NutsSearchCommand addScripts(Collection<String> value)

javascript filter

```
#### ⚙ addScripts(value)  
add javascript filter.
```

NutsSearchCommand addScripts(String value)

javascript filter

```
#### ⚙ clearArchs()  
reset searched for archs
```

NutsSearchCommand clearArchs()

```
#### ⚙ clearIds()  
reset ids to search for
```

NutsSearchCommand clearIds()

```
#### ⚙ clearLockedIds()  
reset locked ids
```

NutsSearchCommand clearLockedIds()

```
#### ⚙ clearPackagings()  
reset packagings to search
```

NutsSearchCommand clearPackagings()

```
#### ⚙ clearRepositories()  
reset repositories to search into
```

NutsSearchCommand clearRepositories()

```
#### ⚙ clearScopes()  
remove all dependency scope filters.
```

NutsSearchCommand clearScopes()

```
#### ⚙ clearScripts()  
remove all javascript filters
```

NutsSearchCommand clearScripts()

```
#### ⚙ configure(skipUnsupported, args)  
configure the current command with the given arguments. This is an  
override of the `` NutsConfigurable#configure(boolean, java.lang.String...)``
```

to help return a more specific return type;

```
NutsSearchCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙ **copy()**

create new instance copy of this

```
NutsSearchCommand copy()
```

⚙ **copyFrom(other)**

copy content from given `other`

```
NutsSearchCommand copyFrom(NutsSearchCommand other)
```

other instance

⚙ **copyFrom(other)**

copy content from given `other`

```
NutsSearchCommand copyFrom(NutsFetchCommand other)
```

other instance

⚙ **copySession()**

copy session

```
NutsSearchCommand copySession()
```

⚙ **getResultClassLoader(parent)**

execute query and return result as class loader

```
ClassLoader getResultClassLoader(ClassLoader parent)
```

parent class loader

⚙ getResultStoreLocations(location)

execute query and return store location path

```
NutsResultList<Path> getResultStoreLocations(NutsStoreLocation location)
```

location type to return

⚙ getResultStrings(columns)

execute query and return the selected columns. Supported columns are :

- all
- long
- status
- install-date
- install-user
- install-folder
- repository
- repository-id
- id
- name
- arch
- packaging
- platform
- os

- osdist
- exec-entry
- file-name
- file
- var-location
- temp-folder
- config-folder
- lib-folder
- log-folder
- cache-folder
- apps-folder

```
NutsResultList<String[]> getResultStrings(String[] columns)
```

columns to return

 **included()**

search for included (in other installations as dependency) packages

```
NutsSearchCommand included()
```

 **installed()**

search for non installed packages

```
NutsSearchCommand installed()
```

 **installedOrIncluded()**

search for non installed or included (in other installations as dependency) packages

NutsSearchCommand **installedOrIncluded()**

⚙ **notInstalled()**

search for non installed packages

NutsSearchCommand **notInstalled()**

⚙ **removeArch(value)**

remove arch to search

NutsSearchCommand **removeArch(String value)**

arch to remove

⚙ **removeld(id)**

remove id to search.

NutsSearchCommand **removeId(String id)**

id to search

⚙ **removeld(id)**

remove id to search.

NutsSearchCommand **removeId(NutsId id)**

id to search

⚙ **removeLockedId(id)**

remove locked ids to prevent them to be updated or the force other ids to use them (the installed

version).

```
NutsSearchCommand removeLockedId(NutsId id)
```

id to unlock

⚙ **removeLockedId(id)**

remove locked ids to prevent them to be updated or the force other ids to use them (the installed version).

```
NutsSearchCommand removeLockedId(String id)
```

id to unlock

⚙ **removePackaging(value)**

remove packaging from search

```
NutsSearchCommand removePackaging(String value)
```

packaging to remove

⚙ **removeRepository(value)**

add repository to search into

```
NutsSearchCommand removeRepository(String value)
```

repository to search into

⚙ **removeScope(scope)**

add dependency scope filter. Only relevant with `#setDependencies(boolean)` and `#setDependenciesTree(boolean)`

NutsSearchCommand **removeScope**(NutsDependencyScope scope)

scope filter

⚙️ removeScope(scope)

remove dependency scope filter. Only relevant with `#setDependencies(boolean)` and `#setDependenciesTree(boolean)`

NutsSearchCommand **removeScope**(NutsDependencyScopePattern scope)

scope filter

⚙️ removeScript(value)

remove javascript filter.

NutsSearchCommand **removeScript**(String value)

javascript filter

⚙️ run()

execute the command and return this instance

NutsSearchCommand **run()**

⚙️ setAnyWhere()

all artifacts (local and remote). If local result found will any way fetch remote.

NutsSearchCommand **setAnyWhere()**

⚙️ `setDefaultLocation()`

unset location to store to fetched id and fall back to default location.

```
NutsSearchCommand setDefaultLocation()
```

⚙️ `setOffline()`

local only (installed or not)

```
NutsSearchCommand setOffline()
```

⚙️ `setOnline()`

local or remote. If local result found will not fetch remote.

```
NutsSearchCommand setOnline()
```

⚙️ `setRemote()`

remote only

```
NutsSearchCommand setRemote()
```

⚙️ `sort(comparator)`

sort results. Comparator should handle types of the result.

```
NutsSearchCommand sort(Comparator comparator)
```

result comparator

⚙️ `toFetch()`

create fetch command initialized with this instance options.

NutsFetchCommand **toFetch()**

⌚ NutsUndeployCommand

```
public net.vpc.app.nuts.NutsUndeployCommand
```

⌚ Instance Properties

✍⌚ ids

```
[read-only] NutsId[] ids  
NutsId[] getIds()
```

✍⌚ offline

```
[read-write] NutsUndeployCommand offline  
boolean isOffline()  
NutsUndeployCommand setOffline(offline)
```

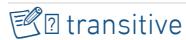
✍⌚ repository

```
[read-write] NutsUndeployCommand repository  
String getRepository()  
NutsUndeployCommand setRepository(repository)
```

⌚⌚ session

update session

```
[write-only] NutsUndeployCommand session  
NutsUndeployCommand setSession(session)
```



```
[read-write] NutsUndeployCommand transitive  
boolean isTransitive()  
NutsUndeployCommand setTransitive(transitive)
```

⚙ Instance Methods

⚙ addId(id)

```
NutsUndeployCommand addId(NutsId id)
```

null

⚙ addId(id)

```
NutsUndeployCommand addId(String id)
```

null

⚙ addIds(value)

```
NutsUndeployCommand addIds(NutsId value)
```

null

⚙ addIds(values)

```
NutsUndeployCommand addIds(String values)
```

null

⚙ clearIds()

NutsUndeployCommand **clearIds()**

⚙️ **configure(skipUnsupported, args)**

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

NutsUndeployCommand **configure(boolean skipUnsupported, String args)**

when true, all unsupported options are skipped argument to configure with

⚙️ **copySession()**

copy session

NutsUndeployCommand **copySession()**

⚙️ **run()**

execute the command and return this instance

NutsUndeployCommand **run()**

☕ **NutsUninstallCommand**

public net.vpc.app.nuts.NutsUninstallCommand

ⓘ **Instance Properties**

📄 ⓘ **args**

[read-only] **String[] args**
String[] getArgs()

 **erase**

```
[read-write] NutsUninstallCommand erase
boolean isErase()
NutsUninstallCommand setErase(erase)
```

 **ids**

```
[read-only] NutsId[] ids
NutsId[] getIds()
```

 **session**

update session

```
[write-only] NutsUninstallCommand session
NutsUninstallCommand setSession(session)
```

 **Instance Methods** **addArg(arg)**

```
NutsUninstallCommand addArg(String arg)
```

null

 **addArgs(args)**

```
NutsUninstallCommand addArgs(Collection<String> args)
```

null

 **addArgs(args)**

```
NutsUninstallCommand addArgs(String args)
```

null

 **addId(id)**

```
NutsUninstallCommand addId(NutsId id)
```

null

 **addId(id)**

```
NutsUninstallCommand addId(String id)
```

null

 **addIds(ids)**

```
NutsUninstallCommand addIds(NutsId ids)
```

null

 **addIds(ids)**

```
NutsUninstallCommand addIds(String ids)
```

null

 **clearArgs()**

```
NutsUninstallCommand clearArgs()
```

⚙️ clearIds()

```
NutsUninstallCommand clearIds()
```

⚙️ configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsUninstallCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ copySession()

copy session

```
NutsUninstallCommand copySession()
```

⚙️ removeld(id)

```
NutsUninstallCommand removeId(NutsId id)
```

null

⚙️ removeld(id)

```
NutsUninstallCommand removeId(String id)
```

null

⚙️ run()

execute the command and return this instance

```
NutsUninstallCommand run()
```

« NutsUpdateOptions

```
public net.vpc.app.nuts.NutsUpdateOptions
```

Generic Add options

author vpc

Constant Fields

serialVersionUID

```
private static final long serialVersionUID = 1
```

Instance Properties

session

current session

```
[read-write] NutsSession public session
private NutsSession session
public NutsSession getSession()
public NutsUpdateOptions setSession(session)
```

« NutsUpdateResult

```
public net.vpc.app.nuts.NutsUpdateResult
```

component update result

Instance Properties

available

return available definition or null

```
[read-only] NutsDefinition available  
NutsDefinition getAvailable()
```

dependencies

return update dependencies

```
[read-only] NutsId[] dependencies  
NutsId[] getDependencies()
```

id

artifact id

```
[read-only] NutsId id  
NutsId getId()
```

local

return installed/local definition or null

```
[read-only] NutsDefinition local  
NutsDefinition getLocal()
```

updateApplied

return true if the update was applied

```
[read-only] boolean updateApplied  
boolean isUpdateApplied()
```

📄 updateAvailable

return true if any update is available. equivalent to ` `isUpdateVersionAvailable() || isUpdateStatusAvailable()`

```
[read-only] boolean updateAvailable  
boolean isUpdateAvailable()
```

📄 updateForced

return true if the update was forced

```
[read-only] boolean updateForced  
boolean isUpdateForced()
```

📄 updateStatusAvailable

return true if artifact has no version update but still have status (default) to be updated

```
[read-only] boolean updateStatusAvailable  
boolean isUpdateStatusAvailable()
```

📄 updateVersionAvailable

return true if artifact has newer available version

```
[read-only] boolean updateVersionAvailable  
boolean isUpdateVersionAvailable()
```

⌚ NutsWorkspaceCommand

```
public net.vpc.app.nuts.NutsWorkspaceCommand
```

Generic Command for usual workspace operations. All Command classes have a 'run' method to perform the operation.

Instance Properties

session

update session

```
[read-write] NutsWorkspaceCommand session  
NutsSession getSession()  
NutsWorkspaceCommand setSession(session)
```

Instance Methods

configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsWorkspaceCommand configure(boolean skipUnsupported, String args)
```

null argument to configure with

copySession()

copy session

```
NutsWorkspaceCommand copySession()
```

run()

execute the command and return this instance

```
NutsWorkspaceCommand run()
```

NutsWorkspaceUpdateResult

```
public net.vpc.app.nuts.NutsWorkspaceUpdateResult
```

Created by vpc on 6/23/17.

Instance Properties

allResults

```
[read-only] NutsUpdateResult[] allResults  
NutsUpdateResult[] getAllResults()
```

allUpdates

```
[read-only] NutsUpdateResult[] allUpdates  
NutsUpdateResult[] getAllUpdates()
```

api

```
[read-only] NutsUpdateResult api  
NutsUpdateResult getApi()
```

artifacts

```
[read-only] NutsUpdateResult[] artifacts  
NutsUpdateResult[] getArtifacts()
```

extensions

```
[read-only] NutsUpdateResult[] extensions  
NutsUpdateResult[] getExtensions()
```

runtime

```
[read-only] NutsUpdateResult runtime  
NutsUpdateResult getRuntime()
```

updateableApi

```
[read-only] boolean updateableApi  
boolean isUpdateableApi()
```

updateableExtensions

```
[read-only] boolean updateableExtensions  
boolean isUpdateableExtensions()
```

updateableRuntime

```
[read-only] boolean updateableRuntime  
boolean isUpdateableRuntime()
```

updateAvailable

```
[read-only] boolean updateAvailable  
boolean isUpdateAvailable()
```

updatesCount

```
[read-only] int updatesCount  
int getUpdatesCount()
```

Config

☕ NutsAddOptions

```
public net.vpc.app.nuts.NutsAddOptions
```

Generic Add options

⌚ Constant Fields

⌚ serialVersionUID

```
private static final long serialVersionUID = 1
```

⌚ Instance Properties

📝 session

current session

```
[read-write] NutsSession public session
private NutsSession session
public NutsSession getSession()
public NutsAddOptions setSession(session)
```

⚙ Instance Methods

⚙ equals(o)

```
boolean equals(Object o)
```

null

⚙ hashCode()

```
int hashCode()
```

⚙️ [toString\(\)](#)

```
String toString()
```

☕️ [NutsAddRepositoryOptions](#)

```
public net.vpc.app.nuts.NutsAddRepositoryOptions
```

repository creation options

⌚️ Constant Fields

⌚️ serialVersionUID

```
private static final long serialVersionUID = 1
```

⌚️ Constructors

⌚️ [NutsAddRepositoryOptions\(\)](#)

default constructor

```
NutsAddRepositoryOptions()
```

⌚️ [NutsAddRepositoryOptions\(other\)](#)

copy constructor

```
NutsAddRepositoryOptions(NutsAddRepositoryOptions other)
```

other

Instance Properties

config

repository config information

```
[read-write] NutsRepositoryConfig public config
private NutsRepositoryConfig config
public NutsRepositoryConfig getConfig()
public NutsAddRepositoryOptions setConfig(value)
```

create

always create. Throw exception if found

```
[read-write] boolean public create
private boolean create
public boolean isCreate()
public NutsAddRepositoryOptions setCreate(value)
```

deployOrder

repository deploy order

```
[read-write] int public deployOrder
private int deployOrder
public int getDeployOrder()
public NutsAddRepositoryOptions setDeployOrder(value)
```

enabled

enabled repository

```
[read-write] boolean public enabled
private boolean enabled
public boolean isEnabled()
public NutsAddRepositoryOptions setEnabled(value)
```

 failSafe

fail safe repository. when fail safe, repository will be ignored if the location is not accessible

```
[read-write] boolean public failSafe  
private boolean failSafe  
public boolean isFailSafe()  
public NutsAddRepositoryOptions setFailSafe(value)
```

 location

repository location

```
[read-write] String public location  
private String location  
public String getLocation()  
public NutsAddRepositoryOptions setLocation(value)
```

 name

repository name (should no include special space or characters)

```
[read-write] String public name  
private String name  
public String getName()  
public NutsAddRepositoryOptions setName(value)
```

 proxy

create a proxy for the created repository

```
[read-write] boolean public proxy  
private boolean proxy  
public boolean isProxy()  
public NutsAddRepositoryOptions setProxy(value)
```

session

current session

```
[read-write] NutsSession public session
private NutsSession session
public NutsSession getSession()
public NutsAddRepositoryOptions setSession(value)
```

temporary

temporary repository

```
[read-write] boolean public temporary
private boolean temporary
public boolean isTemporary()
public NutsAddRepositoryOptions setTemporary(value)
```

Instance Methods

copy()

create a copy of this instance

```
NutsAddRepositoryOptions copy()
```

equals(o)

```
boolean equals(Object o)
```

null

hashCode()

```
int hashCode()
```

 **toString()**

```
String toString()
```

 **NutsCommandAliasConfig**

```
public net.vpc.app.nuts.NutsCommandAliasConfig
```

Command Alias definition class Config

 **Constant Fields** **serialVersionUID**

```
private static final long serialVersionUID = 1
```

 **Instance Properties** **command**

alias command arguments

```
[read-write] String[] public command  
private String[] command  
public String[] getCommand()  
public NutsCommandAliasConfig setCommand(value)
```

 **executorOptions**

alias command execution options

```
[read-write] String[] public executorOptions  
private String[] executorOptions  
public String[] getExecutorOptions()  
public NutsCommandAliasConfig setExecutorOptions(value)
```

factoryId

alias factory id

```
[read-write] String public factoryId  
private String factoryId  
public String getFactoryId()  
public NutsCommandAliasConfig setFactoryId(value)
```

helpCommand

alias help command (command to display help)

```
[read-write] String[] public helpCommand  
private String[] helpCommand  
public String[] getHelpCommand()  
public NutsCommandAliasConfig setHelpCommand(value)
```

helpText

alias help text (meaningful if helpCommand is not defined)

```
[read-write] String public helpText  
private String helpText  
public String getHelpText()  
public NutsCommandAliasConfig setHelpText(value)
```

name

alias name

```
[read-write] String public name  
private String name  
public String getName()  
public NutsCommandAliasConfig setName(value)
```

 owner

alias definition

```
[read-write] NutsId public owner
private NutsId owner
public NutsId getOwner()
public NutsCommandAliasConfig setOwner(value)
```

 Instance Methods equals(o)

```
boolean equals(Object o)
```

null

 hashCode()

```
int hashCode()
```

 toString()

```
String toString()
```

 NutsConfigItem

```
public net.vpc.app.nuts.NutsConfigItem
```

 Constant Fields serialVersionUID

```
private static final long serialVersionUID = 1
```

Instance Properties

configVersion

Api version having created the config

```
[read-write] String public configVersion
private String configVersion = null
public String getConfigVersion()
public void setConfigVersion(configVersion)
```

NutsDefaultWorkspaceOptions

```
public final net.vpc.app.nuts.NutsDefaultWorkspaceOptions
```

Workspace creation/opening options class.

Constant Fields

serialVersionUID

```
private static final long serialVersionUID = 1
```

Static Methods

createHomeLocationKey(storeLocationLayout, location)

creates a string key combining layout and location. le key has the form of a concatenated layout and location ids separated by ':' where null layout is replaced by 'system' keyword. used in `NutsWorkspaceOptions#getHomeLocations()``.

```
String createHomeLocationKey(NutsOsFamily storeLocationLayout, NutsStoreLocation
location)
```

layout location

Constructors

[NutsDefaultWorkspaceOptions\(\)](#)

```
NutsDefaultWorkspaceOptions()
```

[NutsDefaultWorkspaceOptions\(args\)](#)

```
NutsDefaultWorkspaceOptions(String[] args)
```

null

[NutsDefaultWorkspaceOptions\(other\)](#)

```
NutsDefaultWorkspaceOptions(NutsWorkspaceOptions other)
```

null

Instance Properties

 [apiVersion](#)

nuts api version to boot option-type : exported (inherited in child workspaces)

```
[read-write] String public apiVersion  
private String apiVersion = null  
public String getApiVersion\(\)  
public NutsWorkspaceOptionsBuilder setApiVersion\(apiVersion\)
```

 [applicationArguments](#)

option-type : runtime (available only for the current workspace instance)

```
[read-write] String[] public applicationArguments
private String[] applicationArguments
public String[] getApplicationArguments()
public NutsWorkspaceOptionsBuilder setApplicationArguments(applicationArguments)
```

archetype

option-type : create (used when creating new workspace. will not be exported nor promoted to runtime)

```
[read-write] String public archetype
private String archetype
public String getArchetype()
public NutsWorkspaceOptionsBuilder setArchetype(archetype)
```

bootRepositories

option-type : runtime (available only for the current workspace instance)

```
[read-write] String public bootRepositories
private String bootRepositories = null
public String getBootRepositories()
public NutsWorkspaceOptionsBuilder setBootRepositories(bootRepositories)
```

cached

option-type : exported (inherited in child workspaces)

```
[read-write] boolean public cached
private boolean cached = true
public boolean isCached()
public NutsWorkspaceOptionsBuilder setCached(cached)
```

classLoaderSupplier

option-type : runtime (available only for the current workspace instance)

```
[read-write] Supplier<ClassLoader> public classLoaderSupplier
private Supplier<ClassLoader> classLoaderSupplier
public Supplier<ClassLoader> getClassLoaderSupplier()
public NutsWorkspaceOptionsBuilder setClassLoaderSupplier(provider)
```

confirm

option-type : exported (inherited in child workspaces)

```
[read-write] NutsConfirmationMode public confirm
private NutsConfirmationMode confirm = null
public NutsConfirmationMode getConfirm()
public NutsWorkspaceOptionsBuilder setConfirm(confirm)
```

creationTime

option-type : runtime (available only for the current workspace instance)

```
[read-write] long public creationTime
private long creationTime
public long getCreationTime()
public NutsWorkspaceOptionsBuilder setCreationTime(creationTime)
```

credentials

option-type : exported (inherited in child workspaces)

```
[read-write] char[] public credentials
private char[] credentials = null
public char[] getCredentials()
public NutsWorkspaceOptionsBuilder setCredentials(credentials)
```

debug

option-type : runtime (available only for the current workspace instance)

```
[read-write] boolean public debug
private boolean debug = false
public boolean isDebug()
public NutsWorkspaceOptionsBuilder setDebug(debug)
```

dry

if true no real execution, wil dry exec option-type : runtime (available only for the current workspace instance)

```
[read-write] boolean public dry
private boolean dry = false
public boolean isDry()
public NutsWorkspaceOptionsBuilder setDry(dry)
```

excludedExtensions

option-type : exported (inherited in child workspaces)

```
[read-write] String[] public excludedExtensions
private String[] excludedExtensions
public String[] getExcludedExtensions()
public NutsWorkspaceOptionsBuilder setExcludedExtensions(excludedExtensions)
```

excludedRepositories

option-type : exported (inherited in child workspaces)

```
[read-write] String[] public excludedRepositories
private String[] excludedRepositories
public String[] getExcludedRepositories()
public NutsWorkspaceOptionsBuilder setExcludedRepositories(excludedRepositories)
```

executionType

option-type : runtime (available only for the current workspace instance)

```
[read-write] NutsExecutionType public executionType
private NutsExecutionType executionType
public NutsExecutionType getExecutionType()
public NutsWorkspaceOptionsBuilder setExecutionType(executionType)
```

executorOptions

option-type : runtime (available only for the current workspace instance)

```
[read-write] String[] public executorOptions
private String[] executorOptions
public String[] getExecutorOptions()
public NutsWorkspaceOptionsBuilder setExecutorOptions(executorOptions)
```

executorService

not parsed option-type : runtime (available only for the current workspace instance)

```
[read-write] ExecutorService public executorService
private ExecutorService executorService = null
public ExecutorService getExecutorService()
public NutsWorkspaceOptionsBuilder setExecutorService(executorService)
```

fetchStrategy

option-type : exported (inherited in child workspaces)

```
[read-write] NutsFetchStrategy public fetchStrategy
private NutsFetchStrategy fetchStrategy = NutsFetchStrategy.ONLINE
public NutsFetchStrategy getFetchStrategy()
public NutsWorkspaceOptionsBuilder setFetchStrategy(fetchStrategy)
```

global

if true consider global/system repository

option-type : exported (inherited in child workspaces)

```
[read-write] boolean public global
private boolean global
public boolean isGlobal()
public NutsWorkspaceOptionsBuilder setGlobal(global)
```

gui

if true consider GUI/Swing mode

option-type : exported (inherited in child workspaces)

```
[read-write] boolean public gui
private boolean gui
public boolean isGui()
public NutsWorkspaceOptionsBuilder setGui(gui)
```

homeLocations

option-type : create (used when creating new workspace. will not be exported nor promoted to runtime)

```
[read-write] Map<String, String> public homeLocations
private Map<String, String> homeLocations = new HashMap<>()
public Map<String, String> getHomeLocations()
public NutsWorkspaceOptionsBuilder setHomeLocations(homeLocations)
```

indexed

option-type : exported (inherited in child workspaces)

```
[read-write] boolean public indexed
private boolean indexed = true
public boolean isIndexed()
public NutsWorkspaceOptionsBuilder setIndexed(indexed)
```

inherited

option-type : runtime (available only for the current workspace instance)

```
[read-write] boolean public inherited
private boolean inherited = false
public boolean isInherited()
public NutsWorkspaceOptionsBuilder setInherited(inherited)
```

`javaCommand`

option-type : exported (inherited in child workspaces)

```
[read-write] String public javaCommand
private String javaCommand
public String getJavaCommand()
public NutsWorkspaceOptionsBuilder setJavaCommand(javaCommand)
```

`javaOptions`

option-type : exported (inherited in child workspaces)

```
[read-write] String public javaOptions
private String javaOptions
public String getJavaOptions()
public NutsWorkspaceOptionsBuilder setJavaOptions(javaOptions)
```

`logConfig`

option-type : exported (inherited in child workspaces)

```
[read-write] NutsLogConfig public logConfig
private NutsLogConfig logConfig
public NutsLogConfig getLogConfig()
public NutsWorkspaceOptionsBuilder setLogConfig(logConfig)
```

`name`

user friendly workspace name option-type : exported (inherited in child workspaces)

```
[read-write] String public name
private String name = null
public String getName()
public NutsWorkspaceOptionsBuilder setName(workspaceName)
```

openMode

option-type : runtime (available only for the current workspace instance)

```
[read-write] NutsWorkspaceOpenMode public openMode
private NutsWorkspaceOpenMode openMode = NutsWorkspaceOpenMode.OPEN_OR_CREATE
public NutsWorkspaceOpenMode getOpenMode()
public NutsWorkspaceOptionsBuilder setOpenMode(openMode)
```

outputFormat

option-type : exported (inherited in child workspaces)

```
[read-write] NutsOutputFormat public outputFormat
private NutsOutputFormat outputFormat = null
public NutsOutputFormat getOutputFormat()
public NutsWorkspaceOptionsBuilder setOutputFormat(outputFormat)
```

outputFormatOptions

option-type : exported (inherited in child workspaces)

```
[read-write] List<String> public outputFormatOptions
private final List<String> outputFormatOptions = new ArrayList<>()
public String[] getOutputFormatOptions()
public NutsWorkspaceOptionsBuilder setOutputFormatOptions(options)
```

progressOptions

option-type : exported (inherited in child workspaces)

```
[read-write] String public progressOptions
private String progressOptions = null
public String getProgressOptions()
public NutsWorkspaceOptionsBuilder setProgressOptions(progressOptions)
```

readOnly

option-type : exported (inherited in child workspaces)

```
[read-write] boolean public readOnly
private boolean readOnly = false
public boolean isReadOnly()
public NutsWorkspaceOptionsBuilder setReadOnly(readOnly)
```

recover

option-type : runtime (available only for the current workspace instance)

```
[read-write] boolean public recover
private boolean recover = false
public boolean isRecover()
public NutsWorkspaceOptionsBuilder setRecover(recover)
```

repositoryStoreLocationStrategy

option-type : create (used when creating new workspace. will not be exported nor promoted to runtime)

```
[read-write] NutsStoreLocationStrategy public repositoryStoreLocationStrategy
private NutsStoreLocationStrategy repositoryStoreLocationStrategy = null
public NutsStoreLocationStrategy getRepositoryStoreLocationStrategy()
public NutsWorkspaceOptionsBuilder
setRepositoryStoreLocationStrategy(repositoryStoreLocationStrategy)
```

reset

option-type : runtime (available only for the current workspace instance)

```
[read-write] boolean public reset
private boolean reset = false
public boolean isReset()
public NutsWorkspaceOptionsBuilder setReset(reset)
```

`runtimeld`

nuts runtime id (or version) to boot option-type : exported (inherited in child workspaces)

```
[read-write] String public runtimeId
private String runtimeId
public String getRuntimeId()
public NutsWorkspaceOptionsBuilder setRuntimeId(runtimeId)
```

`skipBoot`

if true, do not bootstrap workspace after reset/recover. When reset/recover is not active this option is not accepted and an error will be thrown

```
[read-write] boolean public skipBoot
private boolean skipBoot
public boolean isSkipBoot()
public NutsWorkspaceOptionsBuilder setSkipBoot(skipBoot)
```

`skipCompanions`

if true, do not install nuts companion tools upon workspace creation option-type : exported (inherited in child workspaces)

```
[read-write] boolean public skipCompanions
private boolean skipCompanions
public boolean isSkipCompanions()
public NutsWorkspaceOptionsBuilder setSkipCompanions(skipInstallCompanions)
```

`skipWelcome`

if true, do not run welcome when no application arguments were resolved. defaults to false option-

type : exported (inherited in child workspaces)

```
[read-write] boolean public skipWelcome
private boolean skipWelcome
public boolean isSkipWelcome()
public NutsWorkspaceOptionsBuilder setSkipWelcome(skipWelcome)
```

 stderr

not parsed option-type : runtime (available only for the current workspace instance)

```
[read-write] PrintStream public stderr
private PrintStream stderr = null
public PrintStream getStderr()
public NutsWorkspaceOptionsBuilder setStderr(stderr)
```

 stdin

not parsed option-type : runtime (available only for the current workspace instance)

```
[read-write] InputStream public stdin
private InputStream stdin = null
public InputStream getStdin()
public NutsWorkspaceOptionsBuilder setStdin(stdin)
```

 stdout

not parsed option-type : runtime (available only for the current workspace instance)

```
[read-write] PrintStream public stdout
private PrintStream stdout = null
public PrintStream getStdout()
public NutsWorkspaceOptionsBuilder setStdout(stdout)
```

 storeLocationLayout

option-type : create (used when creating new workspace. will not be exported nor promoted to runtime)

```
[read-write] NutsOsFamily public storeLocationLayout
private NutsOsFamily storeLocationLayout = null
public NutsOsFamily getStoreLocationLayout()
public NutsWorkspaceOptionsBuilder setStoreLocationLayout(storeLocationLayout)
```

storeLocationStrategy

option-type : create (used when creating new workspace. will not be exported nor promoted to runtime)

```
[read-write] NutsStoreLocationStrategy public storeLocationStrategy
private NutsStoreLocationStrategy storeLocationStrategy = null
public NutsStoreLocationStrategy getStoreLocationStrategy()
public NutsWorkspaceOptionsBuilder setStoreLocationStrategy(storeLocationStrategy)
```

storeLocations

option-type : create (used when creating new workspace. will not be exported nor promoted to runtime)

```
[read-write] Map<String, String> public storeLocations
private Map<String, String> storeLocations = new HashMap<>()
public Map<String, String> getStoreLocations()
public NutsWorkspaceOptionsBuilder setStoreLocations(storeLocations)
```

terminalMode

option-type : exported (inherited in child workspaces)

```
[read-write] NutsTerminalMode public terminalMode
private NutsTerminalMode terminalMode = null
public NutsTerminalMode getTerminalMode()
public NutsWorkspaceOptionsBuilder setTerminalMode(terminalMode)
```

 trace

option-type : exported (inherited in child workspaces)

```
[read-write] boolean public trace
private boolean trace = true
public boolean isTrace()
public NutsWorkspaceOptionsBuilder setTrace(trace)
```

 transientRepositories

option-type : exported (inherited in child workspaces)

```
[read-write] String[] public transientRepositories
private String[] transientRepositories
public String[] getTransientRepositories()
public NutsWorkspaceOptionsBuilder
setTransientRepositories(transientRepositories)
```

 transitive

option-type : exported (inherited in child workspaces)

```
[read-write] boolean public transitive
private boolean transitive = true
public boolean isTransitive()
public NutsWorkspaceOptionsBuilder setTransitive(transitive)
```

 userName

option-type : exported (inherited in child workspaces)

```
[read-only] public String userName
private String userName = null
public String getUserName()
```

👤 username

set login

```
[write-only] NutsWorkspaceOptionsBuilder public username  
public NutsWorkspaceOptionsBuilder setUsername(username)
```

📝 workspace

workspace folder location path option-type : exported (inherited in child workspaces)

```
[read-write] String public workspace  
private String workspace = null  
public String getWorkspace()  
public NutsWorkspaceOptionsBuilder setWorkspace(workspace)
```

⚙ Instance Methods

⚙ addOutputFormatOptions(options)

add output format options

```
NutsWorkspaceOptionsBuilder addOutputFormatOptions(String options)
```

new value

⚙ copy()

```
NutsWorkspaceOptionsBuilder copy()
```

⚙ format()

```
NutsWorkspaceOptionsFormat format()
```

⚙️ `getHomeLocation(layout, location)`

```
String getHomeLocation(NutsOsFamily layout, NutsStoreLocation location)
```

null null

⚙️ `getStoreLocation(folder)`

```
String getStoreLocation(NutsStoreLocation folder)
```

null

⚙️ `parse(args)`

parse arguments

```
NutsWorkspaceOptionsBuilder parse(String[] args)
```

arguments

⚙️ `setHomeLocation(layout, location, value)`

set home location

```
NutsWorkspaceOptionsBuilder setHomeLocation(NutsOsFamily layout,  
NutsStoreLocation location, String value)
```

layout location new value

⚙️ `setStoreLocation(location, value)`

set store location

```
NutsWorkspaceOptionsBuilder setStoreLocation(NutsStoreLocation location, String  
value)
```

location new value

 [toString\(\)](#)

```
String toString()
```

 [NutsExtensionInformation](#)

```
public net.vpc.app.nuts.NutsExtensionInformation
```

Extension information

 [Instance Properties](#)

  [author](#)

extension main author(s)

```
[read-only] String author  
String getAuthor()
```

  [category](#)

extension category

```
[read-only] String category  
String getCategory()
```

  [description](#)

extension long description

```
[read-only] String description  
String getDescription()
```

 [id](#)

extension id

```
[read-only] NutsId id  
NutsId getId()
```

 [name](#)

extension user name

```
[read-only] String name  
String getName()
```

 [source](#)

extension source

```
[read-only] String source  
String getSource()
```

 [NutsRepositoryConfig](#)

```
public net.vpc.app.nuts.NutsRepositoryConfig
```

 [Constant Fields](#) [serialVersionUID](#)

```
private static final long serialVersionUID = 1
```

 [Constructors](#)

 [NutsRepositoryConfig\(\)](#)

 [NutsRepositoryConfig\(\)](#)

 [Instance Properties](#)

 [authenticationAgent](#)

```
[read-write] String public authenticationAgent  
private String authenticationAgent  
public String getAuthenticationAgent()  
public NutsRepositoryConfig setAuthenticationAgent(authenticationAgent)
```

 [env](#)

```
[read-write] Map<String, String> public env  
private Map<String, String> env  
public Map<String, String> getEnv()  
public NutsRepositoryConfig setEnv(env)
```

 [groups](#)

```
[read-write] String public groups  
private String groups  
public String getGroups()  
public NutsRepositoryConfig setGroups(groups)
```

 [indexEnabled](#)

```
[read-write] boolean public indexEnabled  
private boolean indexEnabled  
public boolean isIndexEnabled()  
public NutsRepositoryConfig setIndexEnabled(indexEnabled)
```

 location

```
[read-write] String public location
private String location
public String getLocation()
public NutsRepositoryConfig setLocation(location)
```

 mirrors

```
[read-write] List<NutsRepositoryRef> public mirrors
private List<NutsRepositoryRef> mirrors
public List<NutsRepositoryRef> getMirrors()
public NutsRepositoryConfig setMirrors(mirrors)
```

 name

```
[read-write] String public name
private String name
public String getName()
public NutsRepositoryConfig setName(name)
```

 storeLocationStrategy

```
[read-write] NutsStoreLocationStrategy public storeLocationStrategy
private NutsStoreLocationStrategy storeLocationStrategy = null
public NutsStoreLocationStrategy getStoreLocationStrategy()
public NutsRepositoryConfig setStoreLocationStrategy(storeLocationStrategy)
```

 storeLocations

```
[read-write] Map<String, String> public storeLocations
private Map<String, String> storeLocations = null
public Map<String, String> getStoreLocations()
public NutsRepositoryConfig setStoreLocations(storeLocations)
```

type

```
[read-write] String public type
private String type
public String getType()
public NutsRepositoryConfig setType(type)
```

users

```
[read-write] List<NutsUserConfig> public users
private List<NutsUserConfig> users
public List<NutsUserConfig> getUserspublic NutsRepositoryConfig setUsers(users)
```

uuid

```
[read-write] String public uuid
private String uuid
public String getUuid()
public NutsRepositoryConfig setUuid(uuid)
```

Instance Methods

equals(obj)

```
boolean equals(Object obj)
```

null

hashCode()

```
int hashCode()
```

toString()

```
String toString()
```

📄 NutsRepositoryConfigManager

```
public net.vpc.app.nuts.NutsRepositoryConfigManager
```

⌚ Instance Properties

📄⌚ deployOrder

```
[read-only] int deployOrder  
int getDeployOrder()
```

📄⌚ enabled

```
[read-only] boolean enabled  
boolean isEnabled()
```

📄⌚ env

```
[read-only] Map<String, String> env  
Map<String, String> getEnv()
```

📄⌚ globalName

global name is independent from workspace

```
[read-only] String globalName  
String getGlobalName()
```

📄⌚ groups

```
[read-only] String groups  
String getGroups()
```

indexEnabled

```
[read-only] boolean indexEnabled  
boolean isIndexEnabled()
```

indexSubscribed

```
[read-only] boolean indexSubscribed  
boolean isIndexSubscribed()
```

name

name is the name attributed by the containing workspace. It is defined in NutsRepositoryRef

```
[read-only] String name  
String getName()
```

speed

```
[read-only] int speed  
int getSpeed()
```

storeLocation

```
[read-only] Path storeLocation  
Path getStoreLocation()
```

storeLocationStrategy

```
[read-only] NutsStoreLocationStrategy storeLocationStrategy  
NutsStoreLocationStrategy getStoreLocationStrategy()
```

supportedMirroring

```
[read-only] boolean supportedMirroring  
boolean isSupportedMirroring()
```

temporary

```
[read-only] boolean temporary  
boolean isTemporary()
```

type

```
[read-only] String type  
String getType()
```

uuid

```
[read-only] String uuid  
String getUuid()
```

Instance Methods

addMirror(definition)

add new repository

```
NutsRepository addMirror(NutsRepositoryDefinition definition)
```

repository definition

⚙️ **addMirror(options)**

add new repository

```
NutsRepository addMirror(NutsAddRepositoryOptions options)
```

repository definition

⚙️ **findMirror(repositoryIdOrName, session)**

search for (or return null) a repository with the given repository name or id.

```
NutsRepository findMirror(String repositoryIdOrName, NutsSession session)
```

repository name or id session

⚙️ **findMirrorById(repositoryNameOrId, session)**

```
NutsRepository findMirrorById(String repositoryNameOrId, NutsSession session)
```

null null

⚙️ **findMirrorByName(repositoryNameOrId, session)**

```
NutsRepository findMirrorByName(String repositoryNameOrId, NutsSession session)
```

null null

⚙️ **getEnv(inherit)**

```
Map<String, String> getEnv(boolean inherit)
```

null

⚙️ **getEnv(property, defaultValue)**

```
String getEnv(String property, String defaultValue)
```

null null

⚙️ **getEnv(key, defaultValue, inherit)**

```
String getEnv(String key, String defaultValue, boolean inherit)
```

null null null

⚙️ **getLocation(expand)**

return repository configured location as string

```
String getLocation(boolean expand)
```

when true, location will be expanded (~ and \$ params will be expanded)

⚙️ **getMirror(repositoryIdOrName, session)**

search for (or throw error) a repository with the given repository name or id.

```
NutsRepository getMirror(String repositoryIdOrName, NutsSession session)
```

repository name or id session

⚙️ **getMirrors(session)**

```
NutsRepository[] getMirrors(NutsSession session)
```

null

⚙️ `getSpeed(session)`

```
int getSpeed(NutsSession session)
```

null

⚙️ `getStoreLocation(folderType)`

```
Path getStoreLocation(NutsStoreLocation folderType)
```

null

⚙️ `name()`

```
String name()
```

⚙️ `removeMirror(repositoryId, options)`

```
NutsRepositoryConfigManager removeMirror(String repositoryId, NutsRemoveOptions options)
```

repository id pr id remove options

⚙️ `save(session)`

```
void save(NutsSession session)
```

null

⚙️ `save(force, session)`

```
boolean save(boolean force, NutsSession session)
```

null null

⚙️ `setEnabled(enabled, options)`

```
NutsRepositoryConfigManager setEnabled(boolean enabled, NutsUpdateOptions options)
```

null null

⚙️ `setEnv(property, value, options)`

```
void setEnv(String property, String value, NutsUpdateOptions options)
```

null null null

⚙️ `setIndexEnabled(enabled, options)`

```
NutsRepositoryConfigManager setIndexEnabled(boolean enabled, NutsUpdateOptions options)
```

null null

⚙️ `setMirrorEnabled(repoName, enabled, options)`

```
NutsRepositoryConfigManager setMirrorEnabled(String repoName, boolean enabled, NutsUpdateOptions options)
```

null null null

⚙️ `setTemporary(enabled, options)`

```
NutsRepositoryConfigManager setTemporary(boolean enabled, NutsUpdateOptions options)
```

null null

⚙️ **subscribeIndex(session)**

```
NutsRepositoryConfigManager subscribeIndex(NutsSession session)
```

null

⚙️ **unsubscribeIndex(session)**

```
NutsRepositoryConfigManager unsubscribeIndex(NutsSession session)
```

null

⚙️ **uuid()**

```
String uuid()
```

⌚️ **NutsRepositoryDefinition**

```
public net.vpc.app.nuts.NutsRepositoryDefinition
```

⌚️ Constant Fields

⌚️ ORDER_SYSTEM_LOCAL

```
public static final int ORDER_SYSTEM_LOCAL = 2000
```

⌚️ ORDER_USER_LOCAL

```
public static final int ORDER_USER_LOCAL = 1000
```

⌚️ ORDER_USER_REMOTE

```
public static final int ORDER_USER_REMOTE = 10000
```

Constructors

[NutsRepositoryDefinition\(\)](#)

```
NutsRepositoryDefinition()
```

[NutsRepositoryDefinition\(o\)](#)

```
NutsRepositoryDefinition(NutsRepositoryDefinition o)
```

null

Instance Properties

 [create](#)

```
[read-write] boolean public create
private boolean create
public boolean isCreate()
public NutsRepositoryDefinition setCreate(create)
```

 [deployOrder](#)

```
[read-write] int public deployOrder
private int deployOrder = 100
public int getDeployOrder()
public NutsRepositoryDefinition setDeployOrder(deployPriority)
```

 [failSafe](#)

```
[read-write] boolean public failSafe  
private boolean failSafe  
public boolean isFailSafe()  
public NutsRepositoryDefinition setFailSafe(failSafe)
```

location

```
[read-write] String public location  
private String location  
public String getLocation()  
public NutsRepositoryDefinition setLocation(location)
```

name

```
[read-write] String public name  
private String name  
public String getName()  
public NutsRepositoryDefinition setName(name)
```

order

```
[read-write] int public order  
private int order  
public int getOrder()  
public NutsRepositoryDefinition setOrder(order)
```

proxy

```
[read-write] boolean public proxy  
private boolean proxy  
public boolean isProxy()  
public NutsRepositoryDefinition setProxy(proxy)
```

reference

```
[read-write] boolean public reference
private boolean reference
public boolean isReference()
public NutsRepositoryDefinition setReference(reference)
```

session

```
[read-write] NutsSession public session
private NutsSession session
public NutsSession getSession()
public NutsRepositoryDefinition setSession(session)
```

storeLocationStrategy

```
[read-write] NutsStoreLocationStrategy public storeLocationStrategy
private NutsStoreLocationStrategy storeLocationStrategy
public NutsStoreLocationStrategy getStoreLocationStrategy()
public NutsRepositoryDefinition setStoreLocationStrategy(storeLocationStrategy)
```

temporary

```
[read-write] boolean public temporary
private boolean temporary
public boolean isTemporary()
public void setTemporary(temporary)
```

type

```
[read-write] String public type
private String type
public String getType()
public NutsRepositoryDefinition setType(type)
```

Instance Methods

 [copy\(\)](#)

```
NutsRepositoryDefinition copy\(\)
```

 [toString\(\)](#)

```
String toString\(\)
```

 [NutsSdkLocation](#)

```
public net.vpc.app.nuts.NutsSdkLocation
```

SDK location

  [Constant Fields](#)

  [serialVersionUID](#)

```
public static final long serialVersionUID = 2
```

 [Constructors](#)

 [NutsSdkLocation\(id, product, name, path, version, packaging\)](#)

default constructor

```
NutsSdkLocation(NutsId id, String product, String name, String path, String  
version, String packaging)
```

id sdk product. In java this is Oracle JDK or OpenJDK. sdk name sdk path sdk version sdk packaging.
for Java SDK this is room to set JRE or JDK.

 [Instance Properties](#)

 id

```
[read-only] public NutsId id  
private final NutsId id  
public NutsId getId()
```

 name

sdk name

```
[read-only] public String name  
private final String name  
public String getName()
```

 packaging

sdk packaging. for Java SDK this is room to set JRE or JDK.

```
[read-only] public String packaging  
private final String packaging  
public String getPackaging()
```

 path

sdk path

```
[read-only] public String path  
private final String path  
public String getPath()
```

 product

sdk product. In java this is Oracle JDK or OpenJDK.

```
[read-only] public String product  
private final String product  
public String getProduct()
```

📄 version

sdk version

```
[read-only] public String version  
private final String version  
public String getVersion()
```

⚙️ Instance Methods

⚙️ equals(o)

```
boolean equals(Object o)
```

null

⚙️ hashCode()

```
int hashCode()
```

⚙️ toString()

```
String toString()
```

☕ NutsUpdateStatisticsCommand

```
public net.vpc.app.nuts.NutsUpdateStatisticsCommand
```

Instance Properties

session

update session

```
[write-only] NutsUpdateStatisticsCommand session  
NutsUpdateStatisticsCommand setSession(session)
```

Instance Methods

add(repoOrPath)

add path if repoOrPath is a path (contains path separator or is '.' or '..') if not add repo name or id

```
void add(String repoOrPath)
```

repo uuid, name or path

addPath(s)

```
NutsUpdateStatisticsCommand addPath(Path s)
```

null

addPaths(all)

```
NutsUpdateStatisticsCommand addPaths(Path all)
```

null

addPaths(all)

```
NutsUpdateStatisticsCommand addPaths(Collection<Path> all)
```

null

⚙️ **addRepo(s)**

```
NutsUpdateStatisticsCommand addRepo(String s)
```

null

⚙️ **addRepos(all)**

```
NutsUpdateStatisticsCommand addRepos(String all)
```

null

⚙️ **addRepos(all)**

```
NutsUpdateStatisticsCommand addRepos(Collection<String> all)
```

null

⚙️ **clearPaths()**

```
NutsUpdateStatisticsCommand clearPaths()
```

⚙️ **clearRepos()**

```
NutsUpdateStatisticsCommand clearRepos()
```

⚙️ **configure(skipUnsupported, args)**

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsUpdateStatisticsCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙ **copySession()**

copy session

NutsUpdateStatisticsCommand **copySession()**

⚙ **path(s)**

NutsUpdateStatisticsCommand **path(Path s)**

null

⚙ **removePath(s)**

NutsUpdateStatisticsCommand **removePath(Path s)**

null

⚙ **removeRepo(s)**

NutsUpdateStatisticsCommand **removeRepo(String s)**

null

⚙ **repo(s)**

NutsUpdateStatisticsCommand **repo(String s)**

null

⚙ **run()**

execute the command and return this instance

```
NutsUpdateStatisticsCommand run()
```

⌚ NutsUser

```
public net.vpc.app.nuts.NutsUser
```

Effective (including inherited) user information

⌚ Instance Properties

📄⌚ groups

user groups

```
[read-only] String[] groups  
String[] getGroups()
```

📄⌚ inheritedPermissions

user inherited allowed permissions

```
[read-only] String[] inheritedPermissions  
String[] getInheritedPermissions()
```

📄⌚ permissions

user allowed permissions

```
[read-only] String[] permissions  
String[] getPermissions()
```

📄⌚ remoteldentity

return remote identity if applicable

```
[read-only] String remoteIdentity  
String getRemoteIdentity()
```

📄 User

return user name

```
[read-only] String user  
String getUser()
```

⚙️ Instance Methods

⚙️ hasCredentials()

```
true if the user has some credentials
```

```
boolean hasCredentials()
```

☕ NutsUserConfig

```
public final net.vpc.app.nuts.NutsUserConfig
```

⌚ Constant Fields

⌚ serialVersionUID

```
private static final long serialVersionUID = 2
```

⌚ Constructors

⌚ NutsUserConfig()

```
NutsUserConfig()
```

[NutsUserConfig\(other\)](#)

```
NutsUserConfig(NutsUserConfig other)
```

null

[NutsUserConfig\(user, credentials, groups, permissions\)](#)

```
NutsUserConfig(String user, String credentials, String[] groups, String[] permissions)
```

null null null null

[Instance Properties](#)

 [credentials](#)

```
[read-write] String public credentials  
private String credentials  
public String getCredentials()  
public void setCredentials(credentials)
```

 [groups](#)

```
[read-write] String[] public groups  
private String[] groups  
public String[] getGroups()  
public void setGroups(groups)
```

 [permissions](#)

```
[read-write] String[] public permissions  
private String[] permissions  
public String[] getPermissions()  
public void setPermissions(permissions)
```

remoteCredentials

```
[read-write] String public remoteCredentials  
private String remoteCredentials  
public String getRemoteCredentials()  
public void setRemoteCredentials(remoteCredentials)
```

remoteIdentity

```
[read-write] String public remoteIdentity  
private String remoteIdentity  
public String getRemoteIdentity()  
public void setRemoteIdentity(remoteIdentity)
```

user

```
[read-write] String public user  
private String user  
public String getUser()  
public void setUser(user)
```

Instance Methods

equals(o)

```
boolean equals(Object o)
```

null

⚙️ **hashCode()**

```
int hashCode()
```

⚙️ **toString()**

```
String toString()
```

☕️ NutsWorkspaceCommandAlias

```
public net.vpc.app.nuts.NutsWorkspaceCommandAlias
```

⌚ Instance Properties

📄⌚ **command**

```
[read-only] String[] command  
String[] getCommand()
```

📄⌚ **executorOptions**

```
[read-only] String[] executorOptions  
String[] getExecutorOptions()
```

📄⌚ **factoryId**

```
[read-only] String factoryId  
String getFactoryId()
```

📄⌚ **helpText**

```
[read-only] String helpText  
String getHelpText()
```

📄 name

```
[read-only] String name  
String getName()
```

📄 owner

```
[read-only] NutsId owner  
NutsId getOwner()
```

⚙️ Instance Methods

⚙️ dryExec(args, options, session)

```
void dryExec(String[] args, NutsCommandExecOptions options, NutsSession session)
```

null null null

⚙️ exec(args, options, session)

```
void exec(String[] args, NutsCommandExecOptions options, NutsSession session)
```

null null null

🖨️ NutsWorkspaceCommandFactory

```
public net.vpc.app.nuts.NutsWorkspaceCommandFactory
```

🔗 Instance Properties

📄 factoryId

```
[read-only] String factoryId  
String getFactoryId()
```

priority

```
[read-only] int priority  
int getPriority()
```

Instance Methods

configure(config)

```
void configure(NutsCommandAliasFactoryConfig config)
```

null

findCommand(name, workspace)

```
NutsCommandAliasConfig findCommand(String name, NutsWorkspace workspace)
```

null null

findCommands(workspace)

```
List<NutsCommandAliasConfig> findCommands(NutsWorkspace workspace)
```

null

NutsWorkspaceConfigManager

```
public net.vpc.app.nuts.NutsWorkspaceConfigManager
```

Instance Properties

apild

```
[read-only] NutsId apiId  
NutsId getApiId()
```

📄 API Version

```
[read-only] String apiVersion  
String getApiVersion()
```

📄 Architecture

```
[read-only] NutsId arch  
NutsId getArch()
```

📄 Boot Class Loader

```
[read-only] ClassLoader bootClassLoader  
ClassLoader getBootClassLoader()
```

📄 Boot Class World URLs

```
[read-only] URL[] bootClassWorldURLs  
URL[] getBootClassWorldURLs()
```

📄 Boot Repositories

```
[read-only] String bootRepositories  
String getBootRepositories()
```

📄 Creation Finish Time Millis

```
[read-only] long creationFinishTimeMillis  
long getCreationFinishTimeMillis()
```

creationStartTimeMillis

```
[read-only] long creationStartTimeMillis  
long getCreationStartTimeMillis()
```

creationTimeMillis

```
[read-only] long creationTimeMillis  
long getCreationTimeMillis()
```

defaultRepositories

```
[read-only] NutsRepositoryDefinition[] defaultRepositories  
NutsRepositoryDefinition[] getDefaultRepositories()
```

env

```
[read-only] Map<String, String> env  
Map<String, String> getEnv()
```

global

```
[read-only] boolean global  
boolean isGlobal()
```

homeLocations

all home locations key/value map where keys are in the form "osfamily:location" and values are absolute paths.

```
[read-only] Map<String, String> homeLocations  
Map<String, String> getHomeLocations()
```

 imports

```
[read-only] Set<String> imports  
Set<String> getImports()
```

 indexStoreClientFactory

```
[read-only] NutsIndexStoreFactory indexStoreClientFactory  
NutsIndexStoreFactory getIndexStoreClientFactory()
```

 javaCommand

```
[read-only] String javaCommand  
String getJavaCommand()
```

 javaOptions

```
[read-only] String javaOptions  
String getJavaOptions()
```

 name

```
[read-only] String name  
String getName()
```

 options

```
[read-only] NutsWorkspaceOptions options  
NutsWorkspaceOptions getOptions()
```

 os

```
[read-only] NutsId os  
NutsId getOs()
```

[osDist](#)

```
[read-only] NutsId osDist  
NutsId getOsDist()
```

[osFamily](#)

```
[read-only] NutsOsFamily osFamily  
NutsOsFamily getOsFamily()
```

[platform](#)

```
[read-only] NutsId platform  
NutsId getPlatform()
```

[readOnly](#)

```
[read-only] boolean readOnly  
boolean isReadOnly()
```

[repositoryStoreLocationStrategy](#)

```
[read-only] NutsStoreLocationStrategy repositoryStoreLocationStrategy  
NutsStoreLocationStrategy getRepositoryStoreLocationStrategy()
```

[runtimeId](#)

```
[read-only] NutsId runtimeId  
NutsId getRuntimeId()
```

 **sdkTypes**

```
[read-only] String[] sdkTypes  
String[] getSdkTypes()
```

 **storeLocationLayout**

```
[read-only] NutsOsFamily storeLocationLayout  
NutsOsFamily getStoreLocationLayout()
```

 **storeLocationStrategy**

```
[read-only] NutsStoreLocationStrategy storeLocationStrategy  
NutsStoreLocationStrategy getStoreLocationStrategy()
```

 **storeLocations**

all home locations key/value map where keys are in the form "location" and values are absolute paths.

```
[read-only] Map<String, String> storeLocations  
Map<String, String> getStoreLocations()
```

 **uuid**

```
[read-only] String uuid  
String getUuid()
```

 **workspaceLocation**

```
[read-only] Path workspaceLocation  
Path getWorkspaceLocation()
```

⚙ Instance Methods

⚙ addCommandAlias(command, options)

```
boolean addCommandAlias(NutsCommandAliasConfig command, NutsAddOptions options)
```

null null

⚙ addCommandAliasFactory(commandFactory, options)

```
void addCommandAliasFactory(NutsCommandAliasFactoryConfig commandFactory,  
NutsAddOptions options)
```

null null

⚙ addImports(importExpression, options)

```
void addImports(String[] importExpression, NutsAddOptions options)
```

null null

⚙ addRepository(definition)

```
NutsRepository addRepository(NutsRepositoryDefinition definition)
```

null

⚙ addRepository(options)

```
NutsRepository addRepository(NutsAddRepositoryOptions options)
```

null

addRepository(repository, session)

add temporary repository

```
NutsRepository addRepository(NutsRepositoryModel repository, NutsSession session)
```

temporary repository session

addRepository(repositoryNamedUrl, session)

creates a new repository from the given `repositoryNamedUrl` .Accepted `repositoryNamedUrl` values are :

- 'local' : corresponds to a local updatable repository.

will be named 'local'

- 'm2', '.m2', 'maven-local' : corresponds the local maven folder repository. will be named 'local'
- 'maven-central': corresponds the remote maven central repository. will be named 'local'
- 'maven-git', 'vpc-public-maven': corresponds the remote maven vpc-public-maven git folder repository. will be named 'local'
- 'maven-git', 'vpc-public-nuts': corresponds the remote nuts vpc-public-nuts git folder repository. will be named 'local'
- name=uri-or-path : corresponds the given uri. will be named name. Here are some examples:
 - myremote=http://192.168.6.3/folder
 - myremote=/folder/subfolder
 - myremote=c:/folder/subfolder
- uri-or-path : corresponds the given uri. will be named uri's last path component name. Here are some examples:
 - **http://192.168.6.3/folder** : will be named 'folder'
 - myremote=/folder/subfolder : will be named 'folder'
 - myremote=c:/folder/subfolder : will be named 'folder'

```
NutsRepository addRepository(String repositoryNamedUrl, NutsSession session)
```

repositoryNamedUrl session

⚙ **addSdk**(location, options)

```
boolean addSdk(NutsSdkLocation location, NutsAddOptions options)
```

null null

⚙ **createContentFacelid**(id, desc)

```
NutsId createContentFaceId(NutsId id, NutsDescriptor desc)
```

null null

⚙ **createRepository**(options, rootFolder, parentRepository)

```
NutsRepository createRepository(NutsAddRepositoryOptions options, Path  
rootFolder, NutsRepository parentRepository)
```

null null null

⚙ **createWorkspaceListManager**(name, session)

```
NutsWorkspaceListManager createWorkspaceListManager(String name, NutsSession  
session)
```

null null

⚙ **findCommandAlias**(name, session)

```
NutsWorkspaceCommandAlias findCommandAlias(String name, NutsSession session)
```

null null

⚙️ `findCommandAlias(name, forId, forOwner, session)`

return alias definition for given name id and owner.

```
NutsWorkspaceCommandAlias findCommandAlias(String name, NutsId forId, NutsId  
forOwner, NutsSession session)
```

alias name, not null if not null, the alias name should resolve to the given id if not null, the alias name should resolve to the owner session

⚙️ `findCommandAliases(session)`

```
List<NutsWorkspaceCommandAlias> findCommandAliases(NutsSession session)
```

null

⚙️ `findCommandAliases(id, session)`

```
List<NutsWorkspaceCommandAlias> findCommandAliases(NutsId id, NutsSession  
session)
```

null null

⚙️ `findRepository(repositoryIdOrName, session)`

```
NutsRepository findRepository(String repositoryIdOrName, NutsSession session)
```

repository id or name session

⚙️ `findRepositoryById(repositoryIdOrName, session)`

```
NutsRepository findRepositoryById(String repositoryIdOrName, NutsSession session)
```

null null

⚙️ `findRepositoryByName(repositoryIdOrName, session)`

```
NutsRepository findRepositoryByName(String repositoryIdOrName, NutsSession session)
```

null null

⚙️ `findSdk(sdkType, location, session)`

```
NutsSdkLocation findSdk(String sdkType, NutsSdkLocation location, NutsSession session)
```

null null null

⚙️ `findSdkByName(sdkType, locationName, session)`

```
NutsSdkLocation findSdkByName(String sdkType, String locationName, NutsSession session)
```

null null null

⚙️ `findSdkByPath(sdkType, path, session)`

```
NutsSdkLocation findSdkByPath(String sdkType, Path path, NutsSession session)
```

null null null

⚙️ `findSdkByVersion(sdkType, version, session)`

```
NutsSdkLocation findSdkByVersion(String sdkType, String version, NutsSession session)
```

null null null

⚙ `getAvailableArchetypes(session)`

```
Set<String> getAvailableArchetypes(NutsSession session)
```

null

⚙ `getCommandFactories(session)`

```
NutsCommandAliasFactoryConfig[] getCommandFactories(NutsSession session)
```

null

⚙ `getDefaultIdBasedir(id)`

```
String getDefaultIdBasedir(NutsId id)
```

null

⚙ `getDefaultIdContentExtension(packaging)`

```
String getDefaultIdContentExtension(String packaging)
```

null

⚙ `getDefaultIdExtension(id)`

```
String getDefaultIdExtension(NutsId id)
```

null

⚙ `getDefaultIdFilename(id)`

```
String getDefaultIdFilename(NutsId id)
```

null

⚙️ `getEnv(property, defaultValue)`

```
String getEnv(String property, String defaultValue)
```

null null

⚙️ `getHomeLocation(folderType)`

```
Path getHomeLocation(NutsStoreLocation folderType)
```

null

⚙️ `getHomeLocation(layout, location)`

```
Path getHomeLocation(NutsOsFamily layout, NutsStoreLocation location)
```

null null

⚙️ `getRepositories(session)`

```
NutsRepository[] getRepositories(NutsSession session)
```

null

⚙️ `getRepository(repositoryIdOrName, session)`

```
NutsRepository getRepository(String repositoryIdOrName, NutsSession session)
```

null null

⚙️ `getRepositoryRefs(session)`

```
NutsRepositoryRef[] getRepositoryRefs(NutsSession session)
```

null

⚙ **getSdk**(sdkType, requestedVersion, session)

```
NutsSdkLocation getSdk(String sdkType, String requestedVersion, NutsSession session)
```

null null null

⚙ **getSdks**(sdkType, session)

```
NutsSdkLocation[] getSdks(String sdkType, NutsSession session)
```

null null

⚙ **getStoreLocation**(folderType)

```
Path getStoreLocation(NutsStoreLocation folderType)
```

null

⚙ **getStoreLocation**(id, folderType)

```
Path getStoreLocation(String id, NutsStoreLocation folderType)
```

null null

⚙ **getStoreLocation**(id, folderType)

```
Path getStoreLocation(NutsId id, NutsStoreLocation folderType)
```

null null

⚙️ **isSupportedRepositoryType(repositoryType)**

```
boolean isSupportedRepositoryType(String repositoryType)
```

null

⚙️ **name()**

```
String name()
```

⚙️ **options()**

```
NutsWorkspaceOptions options()
```

⚙️ **removeAllImports(options)**

```
void removeAllImports(NutsRemoveOptions options)
```

null

⚙️ **removeCommandAlias(name, options)**

```
boolean removeCommandAlias(String name, NutsRemoveOptions options)
```

null null

⚙️ **removeCommandAliasFactory(name, options)**

```
boolean removeCommandAliasFactory(String name, NutsRemoveOptions options)
```

null null

⚙️ `removeImports(importExpression, options)`

```
void removeImports(String[] importExpression, NutsRemoveOptions options)
```

null null

⚙️ `removeRepository(locationOrRepositoryId, options)`

```
NutsWorkspaceConfigManager removeRepository(String locationOrRepositoryId,  
NutsRemoveOptions options)
```

null null

⚙️ `removeSdk(location, options)`

```
NutsSdkLocation removeSdk(NutsSdkLocation location, NutsRemoveOptions options)
```

null null

⚙️ `resolveRepositoryPath(repositoryLocation)`

```
Path resolveRepositoryPath(String repositoryLocation)
```

null

⚙️ `resolveSdkLocation(sdkType, path, preferredName, session)`

verify if the path is a valid sdk path and return null if not

```
NutsSdkLocation resolveSdkLocation(String sdkType, Path path, String  
preferredName, NutsSession session)
```

sdk type sdk path preferredName session

⚙️ **save(session)**

```
void save(NutsSession session)
```

null

⚙️ **save(force, session)**

save config file if force is activated or non read only and some changes was detected in config file

```
boolean save(boolean force, NutsSession session)
```

when true, save will always be performed session

⚙️ **searchSdkLocations(sdkType, session)**

```
NutsSdkLocation[] searchSdkLocations(String sdkType, NutsSession session)
```

null null

⚙️ **searchSdkLocations(sdkType, path, session)**

```
NutsSdkLocation[] searchSdkLocations(String sdkType, Path path, NutsSession session)
```

null null null

⚙️ **setEnv(property, value, options)**

```
void setEnv(String property, String value, NutsUpdateOptions options)
```

null null null

⚙️ **setHomeLocation(layout, folderType, location, options)**

```
void setHomeLocation(NutsOsFamily layout, NutsStoreLocation folderType, String location, NutsUpdateOptions options)
```

null null null null

⚙ **setImports**(imports, options)

```
void setImports(String[] imports, NutsUpdateOptions options)
```

null null

⚙ **setStoreLocation**(folderType, location, options)

```
void setStoreLocation(NutsStoreLocation folderType, String location, NutsUpdateOptions options)
```

null null null

⚙ **setStoreLocationLayout**(layout, options)

```
void setStoreLocationLayout(NutsOsFamily layout, NutsUpdateOptions options)
```

null null

⚙ **setStoreLocationStrategy**(strategy, options)

```
void setStoreLocationStrategy(NutsStoreLocationStrategy strategy, NutsUpdateOptions options)
```

null null

⚙ **stored()**

```
NutsWorkspaceStoredConfig stored()
```

☕ NutsWorkspaceListConfig

```
public net.vpc.app.nuts.NutsWorkspaceListConfig
```

Class for managing a Workspace list

⌚ Constant Fields

⌚ serialVersionUID

```
private static final long serialVersionUID = 2
```

⌚ Constructors

⌚ NutsWorkspaceListConfig()

```
NutsWorkspaceListConfig()
```

⌚ NutsWorkspaceListConfig(other)

```
NutsWorkspaceListConfig(NutsWorkspaceListConfig other)
```

null

⌚ NutsWorkspaceListConfig(uuid, name)

```
NutsWorkspaceListConfig(String uuid, String name)
```

null null

⌚ Instance Properties

name

```
[read-write] String public name
private String name
public String getName()
public NutsWorkspaceListConfig setName(name)
```

uuid

```
[read-write] String public uuid
private String uuid
public String getUuid()
public NutsWorkspaceListConfig setUuid(uuid)
```

workspaces

```
[read-write] List<NutsWorkspaceLocation> public workspaces
private List<NutsWorkspaceLocation> workspaces
public List<NutsWorkspaceLocation> getWorkspacespublic void setWorkspaces(workspaces)
```

Instance Methods

toString()

```
String toString()
```

NutsWorkspaceListManager

```
public net.vpc.app.nuts.NutsWorkspaceListManager
```

Class for managing a Workspace list

Instance Properties

config

```
[read-write] NutsWorkspaceListManager config  
  NutsWorkspaceListConfig getConfig()  
  NutsWorkspaceListManager setConfig(config)
```

workspaces

```
[read-only] List<NutsWorkspaceLocation> workspaces  
  List<NutsWorkspaceLocation> getWorkspaces()
```

Instance Methods

addWorkspace(path)

```
NutsWorkspace addWorkspace(String path)
```

null

getWorkspaceLocation(uuid)

```
NutsWorkspaceLocation getWorkspaceLocation(String uuid)
```

null

removeWorkspace(name)

```
boolean removeWorkspace(String name)
```

null

save()

```
void save()
```

☕ NutsWorkspaceLocation

```
public net.vpc.app.nuts.NutsWorkspaceLocation
```

Class for managing a Workspace list

⌚ Constant Fields

⌚ serialVersionUID

```
private static final long serialVersionUID = 1
```

Constructors

⌚ NutsWorkspaceLocation()

```
NutsWorkspaceLocation()
```

⌚ NutsWorkspaceLocation(other)

```
NutsWorkspaceLocation(NutsWorkspaceLocation other)
```

null

⌚ NutsWorkspaceLocation(uuid, name, location)

```
NutsWorkspaceLocation(String uuid, String name, String location)
```

null null null

⌚ Instance Properties

enabled

```
[read-write] boolean public enabled
private boolean enabled = true
public boolean isEnabled()
public NutsWorkspaceLocation setEnabled(enabled)
```

location

```
[read-write] String public location
private String location
public String getLocation()
public NutsWorkspaceLocation setLocation(location)
```

name

```
[read-write] String public name
private String name
public String getName()
public NutsWorkspaceLocation setName(name)
```

uuid

```
[read-write] String public uuid
private String uuid
public String getUuid()
public NutsWorkspaceLocation setUuid(uuid)
```

Instance Methods

copy()

```
NutsWorkspaceLocation copy()
```

⚙️ equals(o)

```
boolean equals(Object o)
```

null

⚙️ hashCode()

```
int hashCode()
```

⚙️ toString()

```
String toString()
```

☕ NutsWorkspaceOptions

```
public net.vpc.app.nuts.NutsWorkspaceOptions
```

Workspace options class that holds command argument information.

⌚ Instance Properties

📄⌚ apiVersion

nuts api version to boot.

option-type :** exported (inherited in child

workspaces)

```
[read-only] String apiVersion
String getApiVersion()
```

applicationArguments

application arguments.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] String[] applicationArguments  
String[] getApplicationArguments()
```

archetype

workspace archetype to consider when creating a new workspace.

option-type :** create (used when creating new workspace. will not be

exported nor promoted to runtime)

```
[read-only] String archetype  
String getArchetype()
```

bootRepositories

boot repositories ';' separated

option-type :** runtime (available only for the current workspace instance)

```
[read-only] String bootRepositories  
String getBootRepositories()
```

cached

when true, use cache

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean cached  
boolean isCached()
```

📄 classLoaderSupplier

class loader supplier.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] Supplier<ClassLoader> classLoaderSupplier  
Supplier<ClassLoader> getClassLoaderSupplier()
```

📄 confirm

confirm mode.

option-type :** exported (inherited in child workspaces)

```
[read-only] NutsConfirmationMode confirm  
NutsConfirmationMode getConfirm()
```

📄 creationTime

workspace creation evaluated time.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] long creationTime  
long getCreationTime()
```

📄 credentials

credential needed to log into workspace.

option-type :** exported (inherited in child workspaces)

```
[read-only] char[] credentials  
char[] getCredentials()
```

debug

if true, extra debug information is written to standard output. Particularly, exception stack traces are displayed instead of simpler messages.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] boolean debug  
boolean isDebug()
```

dry

if true no real execution, wil dry exec (execute without side effect).

option-type :** runtime (available only for the current workspace instance)

```
[read-only] boolean dry  
boolean isDry()
```

excludedExtensions

extensions to be excluded when opening the workspace.

option-type :** exported (inherited in child workspaces)

```
[read-only] String[] excludedExtensions  
String[] getExcludedExtensions()
```

excludedRepositories

repository list to be excluded when opening the workspace.

option-type :** exported (inherited in child workspaces)

```
[read-only] String[] excludedRepositories  
String[] getExcludedRepositories()
```

executionType

execution type.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] NutsExecutionType executionType  
NutsExecutionType getExecutionType()
```

executorOptions

extra executor options.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] String[] executorOptions  
String[] getExecutorOptions()
```

executorService

executor service used to create worker threads. when null, use default. this option cannot be defined via arguments.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] ExecutorService executorService  
ExecutorService getExecutorService()
```

fetchStrategy

default fetch strategy

option-type :** exported (inherited in child workspaces)

```
[read-only] NutsFetchStrategy fetchStrategy  
NutsFetchStrategy getFetchStrategy()
```

  global

if true consider global/system repository

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean global  
boolean isGlobal()
```

  gui

if true consider GUI/Swing mode

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean gui  
boolean isGui()
```

  homeLocations

return home locations.

option-type :** create (used when creating new workspace. will not be

exported nor promoted to runtime).

```
[read-only] Map<String, String> homeLocations  
Map<String, String> getHomeLocations()
```

indexed

when true, use index

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean indexed  
boolean isIndexed()
```

inherited

if true, workspace were invoked from parent process and hence inherits its options.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] boolean inherited  
boolean isInherited()
```

javaCommand

java command (or java home) used to run workspace.

option-type :** exported (inherited in child workspaces)

```
[read-only] String javaCommand  
String getJavaCommand()
```

javaOptions

java options used to run workspace.

option-type :** exported (inherited in child workspaces)

```
[read-only] String javaOptions  
String getJavaOptions()
```

logConfig

workspace log configuration.

option-type :** exported (inherited in child workspaces)

```
[read-only] NutsLogConfig logConfig  
NutsLogConfig getLogConfig()
```

name

user friendly workspace name.

option-type :** exported (inherited in child

workspaces)

```
[read-only] String name  
String getName()
```

openMode

mode used to open workspace.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] NutsWorkspaceOpenMode openMode  
NutsWorkspaceOpenMode getOpenMode()
```

outputFormat

default output format type.

option-type :** exported (inherited in child workspaces)

```
[read-only] NutsOutputFormat outputFormat  
NutsOutputFormat getOutputFormat()
```

📄💡 **outputFormatOptions**

default output formation options.

option-type :** exported (inherited in child workspaces)

```
[read-only] String[] outputFormatOptions  
String[] getOutputFormatOptions()
```

📄💡 **progressOptions**

return progress options string. progress options configures how progress monitors are processed.
'no' value means that progress is disabled.

option-type :** exported (inherited in child workspaces)

```
[read-only] String progressOptions  
String getProgressOptions()
```

📄💡 **readOnly**

if true, workspace configuration are non modifiable. However cache stills modifiable so that it is possible to load external libraries.

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean readOnly  
boolean isReadOnly()
```

📄💡 **recover**

if true, boot, cache and temp folder are deleted.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] boolean recover  
boolean isRecover()
```

repositoryStoreLocationStrategy

repository store location strategy to consider when creating new repositories for a new workspace.

option-type :** create (used when creating new workspace. will not be
exported nor promoted to runtime)

```
[read-only] NutsStoreLocationStrategy repositoryStoreLocationStrategy  
NutsStoreLocationStrategy getRepositoryStoreLocationStrategy()
```

reset

if true, workspace will be reset (all configuration and runtime files deleted).

option-type :** runtime (available only for the current workspace instance)

```
[read-only] boolean reset  
boolean isReset()
```

runtimeld

nuts runtime id (or version) to boot.

option-type :** exported (inherited in child workspaces)

```
[read-only] String runtimeId  
String getRuntimeId()
```

📄 skipBoot

if true, do not bootstrap workspace after reset/recover. When reset/recover is not active this option is not accepted and an error will be thrown

defaults to false.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] boolean skipBoot  
boolean isSkipBoot()
```

📄 skipCompanions

if true, do not install nuts companion tools upon workspace creation.

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean skipCompanions  
boolean isSkipCompanions()
```

📄 skipWelcome

if true, do not run welcome when no application arguments were resolved.

defaults to false.

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean skipWelcome  
boolean isSkipWelcome()
```

📄 stderr

default standard error. when null, use `System.err` this option cannot be defined via arguments.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] PrintStream stderr  
PrintStream getStderr()
```

stdin

default standard input. when null, use `System.in` this option cannot be defined via arguments.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] InputStream stdin  
InputStream getStdin()
```

stdout

default standard output. when null, use `System.out` this option cannot be defined via arguments.

option-type :** runtime (available only for the current workspace instance)

```
[read-only] PrintStream stdout  
PrintStream getStdout()
```

storeLocationLayout

store location layout to consider when creating a new workspace.

option-type :** create (used when creating new workspace. will not be

exported nor promoted to runtime)

```
[read-only] NutsOsFamily storeLocationLayout  
NutsOsFamily getStoreLocationLayout()
```

📄 storeLocationStrategy

store location strategy for creating a new workspace.

option-type :** create (used when creating new workspace. will not be

exported nor promoted to runtime)

```
[read-only] NutsStoreLocationStrategy storeLocationStrategy  
NutsStoreLocationStrategy getStoreLocationStrategy()
```

📄 storeLocations

store locations map to consider when creating a new workspace.

option-type :** create (used when creating new workspace. will not be

exported nor promoted to runtime)

```
[read-only] Map<String, String> storeLocations  
Map<String, String> getStoreLocations()
```

📄 terminalMode

terminal mode (inherited, formatted, filtered) to use.

option-type :** exported (inherited in child workspaces)

```
[read-only] NutsTerminalMode terminalMode  
NutsTerminalMode getTerminalMode()
```

📄 trace

when true, extra trace user-friendly information is written to standard output.

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean trace  
boolean isTrace()
```

transientRepositories

repositories to register temporarily when running the workspace.

option-type :** exported (inherited in child workspaces)

```
[read-only] String[] transientRepositories  
String[] getTransientRepositories()
```

transitive

when true, use transitive repositories

option-type :** exported (inherited in child workspaces)

```
[read-only] boolean transitive  
boolean isTransitive()
```

userName

username to log into when running workspace.

option-type :** exported (inherited in child workspaces)

```
[read-only] String userName  
String getUserName()
```

workspace

workspace folder location path.

option-type :** exported (inherited in child

workspaces)

```
[read-only] String workspace  
String getWorkspace()
```

⚙ Instance Methods

⚙ **copy()**

create a mutable copy of this instance

```
NutsWorkspaceOptionsBuilder copy()
```

⚙ **format()**

create a new instance of options formatter that help formatting this instance.

```
NutsWorkspaceOptionsFormat format()
```

⚙ **getHomeLocation(layout, location)**

return home location.

option-type :** create (used when creating new workspace. will not be

exported nor promoted to runtime).

```
String getHomeLocation(NutsOsFamily layout, NutsStoreLocation location)
```

layout location

⚙ **getStoreLocation(folder)**

store location for the given folder.

option-type :** create (used when creating new workspace. will not be
exported nor promoted to runtime)

```
String getStoreLocation(NutsStoreLocation folder)
```

folder type

↳ NutsWorkspaceOptionsBuilder

```
public net.vpc.app.nuts.NutsWorkspaceOptionsBuilder
```

Mutable Workspace options

Instance Properties

apiVersion

```
[write-only] NutsWorkspaceOptionsBuilder apiVersion  
NutsWorkspaceOptionsBuilder setApiVersion(apiVersion)
```

applicationArguments

```
[write-only] NutsWorkspaceOptionsBuilder applicationArguments  
NutsWorkspaceOptionsBuilder setApplicationArguments(applicationArguments)
```

archetype

```
[write-only] NutsWorkspaceOptionsBuilder archetype  
NutsWorkspaceOptionsBuilder setArchetype(archetype)
```

bootRepositories

```
[write-only] NutsWorkspaceOptionsBuilder bootRepositories  
NutsWorkspaceOptionsBuilder setBootRepositories(bootRepositories)
```

[\[\] cached](#)

```
[write-only] NutsWorkspaceOptionsBuilder cached  
NutsWorkspaceOptionsBuilder setCached(cached)
```

[\[\] classLoaderSupplier](#)

```
[write-only] NutsWorkspaceOptionsBuilder classLoaderSupplier  
NutsWorkspaceOptionsBuilder setClassLoaderSupplier(provider)
```

[\[\] confirm](#)

```
[write-only] NutsWorkspaceOptionsBuilder confirm  
NutsWorkspaceOptionsBuilder setConfirm(confirm)
```

[\[\] creationTime](#)

```
[write-only] NutsWorkspaceOptionsBuilder creationTime  
NutsWorkspaceOptionsBuilder setCreationTime(creationTime)
```

[\[\] credentials](#)

```
[write-only] NutsWorkspaceOptionsBuilder credentials  
NutsWorkspaceOptionsBuilder setCredentials(credentials)
```

[\[\] debug](#)

```
[write-only] NutsWorkspaceOptionsBuilder debug  
NutsWorkspaceOptionsBuilder setDebug(debug)
```

[\[\]](#) dry

```
[write-only] NutsWorkspaceOptionsBuilder dry  
NutsWorkspaceOptionsBuilder setDry(dry)
```

[\[\]](#) excludedExtensions

```
[write-only] NutsWorkspaceOptionsBuilder excludedExtensions  
NutsWorkspaceOptionsBuilder setExcludedExtensions(excludedExtensions)
```

[\[\]](#) excludedRepositories

```
[write-only] NutsWorkspaceOptionsBuilder excludedRepositories  
NutsWorkspaceOptionsBuilder setExcludedRepositories(excludedRepositories)
```

[\[\]](#) executionType

```
[write-only] NutsWorkspaceOptionsBuilder executionType  
NutsWorkspaceOptionsBuilder setExecutionType(executionType)
```

[\[\]](#) executorOptions

```
[write-only] NutsWorkspaceOptionsBuilder executorOptions  
NutsWorkspaceOptionsBuilder setExecutorOptions(executorOptions)
```

[\[\]](#) executorService

```
[write-only] NutsWorkspaceOptionsBuilder executorService  
NutsWorkspaceOptionsBuilder setExecutorService(executorService)
```

[\[\]](#) fetchStrategy

```
[write-only] NutsWorkspaceOptionsBuilder fetchStrategy  
NutsWorkspaceOptionsBuilder setFetchStrategy(fetchStrategy)
```

[global](#)

```
[write-only] NutsWorkspaceOptionsBuilder global  
NutsWorkspaceOptionsBuilder setGlobal(global)
```

[gui](#)

```
[write-only] NutsWorkspaceOptionsBuilder gui  
NutsWorkspaceOptionsBuilder setGui(gui)
```

[homeLocations](#)

set home locations.

option-type :** create (used when creating new workspace. will not be
exported nor promoted to runtime).

```
[write-only] NutsWorkspaceOptionsBuilder homeLocations  
NutsWorkspaceOptionsBuilder setHomeLocations(homeLocations)
```

[indexed](#)

```
[write-only] NutsWorkspaceOptionsBuilder indexed  
NutsWorkspaceOptionsBuilder setIndexed(indexed)
```

[inherited](#)

```
[write-only] NutsWorkspaceOptionsBuilder inherited  
NutsWorkspaceOptionsBuilder setInherited(inherited)
```

[\[\]](#) `javaCommand`

```
[write-only] NutsWorkspaceOptionsBuilder javaCommand  
NutsWorkspaceOptionsBuilder setJavaCommand(javaCommand)
```

[\[\]](#) `javaOptions`

```
[write-only] NutsWorkspaceOptionsBuilder javaOptions  
NutsWorkspaceOptionsBuilder setJavaOptions(javaOptions)
```

[\[\]](#) `logConfig`

```
[write-only] NutsWorkspaceOptionsBuilder logConfig  
NutsWorkspaceOptionsBuilder setLogConfig(logConfig)
```

[\[\]](#) `name`

```
[write-only] NutsWorkspaceOptionsBuilder name  
NutsWorkspaceOptionsBuilder setName(workspaceName)
```

[\[\]](#) `openMode`

```
[write-only] NutsWorkspaceOptionsBuilder openMode  
NutsWorkspaceOptionsBuilder setOpenMode(openMode)
```

[\[\]](#) `outputFormat`

```
[write-only] NutsWorkspaceOptionsBuilder outputFormat  
NutsWorkspaceOptionsBuilder setOutputFormat(outputFormat)
```

[\[\]](#) `outputFormatOptions`

```
[write-only] NutsWorkspaceOptionsBuilder outputFormatOptions  
NutsWorkspaceOptionsBuilder setOutputFormatOptions(options)
```

[progressOptions](#)

```
[write-only] NutsWorkspaceOptionsBuilder progressOptions  
NutsWorkspaceOptionsBuilder setProgressOptions(progressOptions)
```

[readOnly](#)

```
[write-only] NutsWorkspaceOptionsBuilder readOnly  
NutsWorkspaceOptionsBuilder setReadOnly(readOnly)
```

[recover](#)

```
[write-only] NutsWorkspaceOptionsBuilder recover  
NutsWorkspaceOptionsBuilder setRecover(recover)
```

[repositoryStoreLocationStrategy](#)

```
[write-only] NutsWorkspaceOptionsBuilder repositoryStoreLocationStrategy  
NutsWorkspaceOptionsBuilder  
setRepositoryStoreLocationStrategy(repositoryStoreLocationStrategy)
```

[reset](#)

```
[write-only] NutsWorkspaceOptionsBuilder reset  
NutsWorkspaceOptionsBuilder setReset(reset)
```

[runtimeld](#)

```
[write-only] NutsWorkspaceOptionsBuilder runtimeId  
NutsWorkspaceOptionsBuilder setRuntimeId(runtimeId)
```

[skipBoot](#)

if true, do not bootstrap workspace after reset/recover. When reset/recover is not active this option is not accepted and an error will be thrown

defaults to false.

option-type :** runtime (available only for the current workspace instance)

```
[write-only] NutsWorkspaceOptionsBuilder skipBoot  
NutsWorkspaceOptionsBuilder setSkipBoot(skipBoot)
```

[skipCompanions](#)

```
[write-only] NutsWorkspaceOptionsBuilder skipCompanions  
NutsWorkspaceOptionsBuilder setSkipCompanions(skipInstallCompanions)
```

[skipWelcome](#)

```
[write-only] NutsWorkspaceOptionsBuilder skipWelcome  
NutsWorkspaceOptionsBuilder setSkipWelcome(skipWelcome)
```

[stderr](#)

```
[write-only] NutsWorkspaceOptionsBuilder stderr  
NutsWorkspaceOptionsBuilder setStderr(stderr)
```

[stdin](#)

```
[write-only] NutsWorkspaceOptionsBuilder stdin  
NutsWorkspaceOptionsBuilder setStdin(stdin)
```

[stdout](#)

```
[write-only] NutsWorkspaceOptionsBuilder stdout  
NutsWorkspaceOptionsBuilder setStdout(stdout)
```

[storeLocationLayout](#)

```
[write-only] NutsWorkspaceOptionsBuilder storeLocationLayout  
NutsWorkspaceOptionsBuilder setStoreLocationLayout(storeLocationLayout)
```

[storeLocationStrategy](#)

```
[write-only] NutsWorkspaceOptionsBuilder storeLocationStrategy  
NutsWorkspaceOptionsBuilder setStoreLocationStrategy(storeLocationStrategy)
```

[storeLocations](#)

set store location strategy for creating a new workspace.

option-type :** create (used when creating new workspace. will not be
exported nor promoted to runtime)

```
[write-only] NutsWorkspaceOptionsBuilder storeLocations  
NutsWorkspaceOptionsBuilder setStoreLocations(storeLocations)
```

[terminalMode](#)

```
[write-only] NutsWorkspaceOptionsBuilder terminalMode  
NutsWorkspaceOptionsBuilder setTerminalMode(terminalMode)
```

[trace](#)

```
[write-only] NutsWorkspaceOptionsBuilder trace  
NutsWorkspaceOptionsBuilder setTrace(trace)
```

Transient Repositories

```
[write-only] NutsWorkspaceOptionsBuilder transientRepositories  
NutsWorkspaceOptionsBuilder setTransientRepositories(transientRepositories)
```

Transitive

```
[write-only] NutsWorkspaceOptionsBuilder transitive  
NutsWorkspaceOptionsBuilder setTransitive(transitive)
```

Username

```
[write-only] NutsWorkspaceOptionsBuilder username  
NutsWorkspaceOptionsBuilder setUsername(username)
```

Workspace

```
[write-only] NutsWorkspaceOptionsBuilder workspace  
NutsWorkspaceOptionsBuilder setWorkspace(workspace)
```

Instance Methods

addOutputFormatOptions(options)

```
NutsWorkspaceOptionsBuilder addOutputFormatOptions(String options)
```

null

setHomeLocation(layout, location, value)

```
NutsWorkspaceOptionsBuilder setHomeLocation(NutsOsFamily layout,  
NutsStoreLocation location, String value)
```

null null null

⚙️ setStoreLocation(location, value)

```
NutsWorkspaceOptionsBuilder setStoreLocation(NutsStoreLocation location, String  
value)
```

null null

🖨️ NutsWorkspaceStoredConfig

```
public net.vpc.app.nuts.NutsWorkspaceStoredConfig
```

Nuts read-only configuration

🔗 Instance Properties

📄 ↴ apild

```
[read-only] NutsId apild  
NutsId getApiId()
```

📄 ↴ bootRepositories

```
[read-only] String bootRepositories  
String getBootRepositories()
```

📄 ↴ global

```
[read-only] boolean global  
boolean isGlobal()
```

📄 ↴ homeLocations

all home locations key/value map where keys are in the form "osfamily:location" and values are absolute paths.

```
[read-only] Map<String, String> homeLocations  
Map<String, String> getHomeLocations()
```

`javaCommand`

```
[read-only] String javaCommand  
String getJavaCommand()
```

`javaOptions`

```
[read-only] String javaOptions  
String getJavaOptions()
```

`name`

```
[read-only] String name  
String getName()
```

`repositoryStoreLocationStrategy`

```
[read-only] NutsStoreLocationStrategy repositoryStoreLocationStrategy  
NutsStoreLocationStrategy getRepositoryStoreLocationStrategy()
```

`runtimeDependencies`

```
[read-only] String runtimeDependencies  
String getRuntimeDependencies()
```

`runtimeld`

```
[read-only] NutsId runtimeId  
NutsId getRuntimeId()
```

📄 storeLocationLayout

```
[read-only] NutsOsFamily storeLocationLayout  
NutsOsFamily getStoreLocationLayout()
```

📄 storeLocationStrategy

```
[read-only] NutsStoreLocationStrategy storeLocationStrategy  
NutsStoreLocationStrategy getStoreLocationStrategy()
```

📄 storeLocations

all home locations key/value map where keys are in the form "location" and values are absolute paths.

```
[read-only] Map<String, String> storeLocations  
Map<String, String> getStoreLocations()
```

⚙️ Instance Methods

⚙️ getHomeLocation(layout, location)

```
String getHomeLocation(NutsOsFamily layout, NutsStoreLocation location)
```

null null

⚙️ getStoreLocation(folderType)

```
String getStoreLocation(NutsStoreLocation folderType)
```

null

Constants

☕ NutsConstants

```
public final net.vpc.app.nuts.NutsConstants
```

Common Nuts constants. Represents various constants used in runtime implementation.

Created by vpc on 1/14/17.

Constructors

⌚ NutsConstants()

private constructor

```
NutsConstants()
```

Descriptor

⌚ NutsClassifierMapping

```
public net.vpc.app.nuts.NutsClassifierMapping
```

classifier selector immutable class. Nuts can select artifact classifier according to filters based on arch, os, os dist and platform. This class defines the mapping to classifier to consider if all the filters. When multiple selectors match, the first one prevails.

⌚ Instance Properties

⌚ arch

arch list filter. at least one of the list must match.

```
[read-only] String[] arch  
String[] getArch()
```

⌚ classifier

classifier to select

```
[read-only] String classifier  
String getClassifier()
```

⌚ os

os list filter. at least one of the list must match.

```
[read-only] String[] os  
String[] getOs()
```

⌚ osdist

os distribution list filter. at least one of the list must match.

```
[read-only] String[] osdist  
String[] getOsdist()
```

packaging

packaging to select

```
[read-only] String packaging  
String getPackaging()
```

platform

platform list filter. al least one of the list must match.

```
[read-only] String[] platform  
String[] getPlatform()
```

NutsClassifierMappingBuilder

```
public net.vpc.app.nuts.NutsClassifierMappingBuilder
```

classifier selector builder class. Nuts can select artifact classifier according to filters based on arch, os, os dist and platform. This class defines the mapping to classifier to consider if all the filters. When multiple selectors match, the first on prevails.

Instance Properties

arch

set archs

```
[read-write] NutsClassifierMappingBuilder arch  
String[] getArch()  
NutsClassifierMappingBuilder setArch(value)
```

 classifier

set classifier

```
[read-write] NutsClassifierMappingBuilder classifier
String getClassifier()
NutsClassifierMappingBuilder setClassifier(value)
```

 os

set oses

```
[read-write] NutsClassifierMappingBuilder os
String[] getOs()
NutsClassifierMappingBuilder setOs(value)
```

 osdist

set os dists

```
[read-write] NutsClassifierMappingBuilder osdist
String[] getOsdist()
NutsClassifierMappingBuilder setOsdist(value)
```

 packaging

set packaging

```
[read-write] NutsClassifierMappingBuilder packaging
String getPackaging()
NutsClassifierMappingBuilder setPackaging(value)
```

 platform

set platforms

```
[read-write] NutsClassifierMappingBuilder platform  
String[] getPlatform()  
NutsClassifierMappingBuilder setPlatform(value)
```

⚙ Instance Methods

⚙ **build()**

create new instance of `NutsClassifierMapping` initialized with this builder's values.

```
NutsClassifierMapping build()
```

⚙ **clear()**

clear all values / reset builder

```
NutsClassifierMappingBuilder clear()
```

⚙ **set(value)**

copy all values from the given builder

```
NutsClassifierMappingBuilder set(NutsClassifierMappingBuilder value)
```

builder to copy from

⚙ **set(value)**

copy all values from the given instance

```
NutsClassifierMappingBuilder set(NutsClassifierMapping value)
```

instance to copy from

 NutsContent

```
public net.vpc.app.nuts.NutsContent
```

Content describes a artifact file location and its characteristics.

 Instance Properties cached

when true, the content was retrieved from cache rather then from remote location.

```
[read-only] boolean cached  
boolean isCached()
```

 path

artifact local path

```
[read-only] Path path  
Path getPath()
```

 temporary

when true, the path location is temporary and should be deleted after usage

```
[read-only] boolean temporary  
boolean isTemporary()
```

 NutsDefaultContent

```
public net.vpc.app.nuts.NutsDefaultContent
```

Default Content implementation.

Instance Fields

file

```
private final Path file
```

Constructors

NutsDefaultContent(file, cached, temporary)

Default Content implementation constructor

```
NutsDefaultContent(Path file, boolean cached, boolean temporary)
```

content file path true if the file is cached (may be not up to date) true if file is temporary (should be deleted later)

Instance Properties

cached

true if the file is cached (may be not up to date)

```
[read-only] public boolean cached  
private final boolean cached  
public boolean isCached()
```

path

content path location

```
[read-only] public Path path  
public Path getPath()
```

📄Temporary

true if file is temporary (should be deleted later)

```
[read-only] public boolean temporary  
private final boolean temporary  
public boolean isTemporary()
```

⚙️ Instance Methods

⚙️ equals(o)

```
boolean equals(Object o)
```

null

⚙️ hashCode()

```
int hashCode()
```

⚙️ toString()

```
String toString()
```

📥 NutsDependency

```
public net.vpc.app.nuts.NutsDependency
```

NutsDependency is an immutable object that contains all information about a component's dependency.

=====

Instance Properties

📄artifactId

return artifact id (aka artifactId)

```
[read-only] String artifactId  
String getArtifactId()
```

classifier

get classifier string value (may be \$ var)

```
[read-only] String classifier  
String getClassifier()
```

exclusions

dependency exclusions

```
[read-only] NutsId[] exclusions  
NutsId[] getExclusions()
```

fullName

return dependency full name in the form namespace://group:name#version?scope=<scope> ` &` optional=<optional>

```
[read-only] String fullName  
String getFullName()
```

groupId

return artifact group id (aka groupId in maven)

```
[read-only] String groupId  
String getGroupId()
```

id

convert to NutsId

```
[read-only] NutsId id  
NutsId getId()
```

longName

return dependency full name in the form group:name#version

```
[read-only] String longName  
String getLongName()
```

namespace

return namespace

```
[read-only] String namespace  
String getNamespace()
```

optional

Indicates the dependency is optional for use of this library.

```
[read-only] String optional  
String getOptional()
```

properties

properties in the url query form

```
[read-only] Map<String, String> properties  
Map<String, String> getProperties()
```

propertiesQuery

properties in the url query form

```
[read-only] String propertiesQuery  
String getPropertiesQuery()
```

scope

get scope string value (may be \$ var).

```
[read-only] String scope  
String getScope()
```

simpleName

return dependency full name in the form group:name

```
[read-only] String simpleName  
String getSimpleName()
```

version

return dependency version

```
[read-only] NutsVersion version  
NutsVersion getVersion()
```

Instance Methods

builder()

return mutable id builder instance initialized with `this` instance.

```
NutsDependencyBuilder builder()
```

NutsDependencyBuilder

```
public net.vpc.app.nuts.NutsDependencyBuilder
```

Dependency Builder (mutable). User should use available 'set' method and finally call `#build()` to get an instance of immutable NutsDependency

Instance Properties

artifactId

set name value

```
[read-write] NutsDependencyBuilder artifactId  
String getArtifactId()  
NutsDependencyBuilder setArtifactId(artifactId)
```

classifier

set classifier value

```
[read-write] NutsDependencyBuilder classifier  
String getClassifier()  
NutsDependencyBuilder setClassifier(classifier)
```

dependency

reset this instance with value

```
[write-only] NutsDependencyBuilder dependency  
NutsDependencyBuilder setDependency(value)
```

exclusions

set exclusions value

```
[read-write] NutsDependencyBuilder exclusions  
NutsId[] getExclusions()  
NutsDependencyBuilder setExclusions(exclusions)
```

fullName

return full name

```
[read-only] String fullName  
String getFullName()
```

groupId

set group value

```
[read-write] NutsDependencyBuilder groupId  
String getGroupId()  
NutsDependencyBuilder setGroupId(groupId)
```

id

set id value

```
[read-write] NutsDependencyBuilder id  
NutsId getId()  
NutsDependencyBuilder setId(id)
```

namespace

set namespace value

```
[read-write] NutsDependencyBuilder namespace  
String getNamespace()  
NutsDependencyBuilder setNamespace(namespace)
```

 optional

set optional value

```
[read-write] NutsDependencyBuilder optional  
String getOptional()  
NutsDependencyBuilder setOptional(optional)
```

 properties

```
[read-write] NutsDependencyBuilder properties  
Map<String, String> getProperties()  
NutsDependencyBuilder setProperties(propertiesQuery)
```

 propertiesQuery

```
[read-only] String propertiesQuery  
String getPropertiesQuery()
```

 scope

set scope value

```
[read-write] NutsDependencyBuilder scope  
String getScope()  
NutsDependencyBuilder setScope(scope)
```

 version

set version value

```
[read-write] NutsDependencyBuilder version  
NutsVersion getVersion()  
NutsDependencyBuilder setVersion(version)
```

⚙ Instance Methods

⚙ addProperties(propertiesQuery)

```
NutsDependencyBuilder addProperties(String propertiesQuery)
```

null

⚙ addProperties(queryMap)

```
NutsDependencyBuilder addProperties(Map<String, String> queryMap)
```

null

⚙ build()

build new instance of NutsDependencies

```
NutsDependency build()
```

⚙ clear()

reset this instance

```
NutsDependencyBuilder clear()
```

⚙ set(value)

reset this instance with value

```
NutsDependencyBuilder set(NutsDependencyBuilder value)
```

new value

⚙️ set(value)

reset this instance with value

```
NutsDependencyBuilder set(NutsDependency value)
```

new value

⚙️ setProperty(property, value)

```
NutsDependencyBuilder setProperty(String property, String value)
```

null null

☕ NutsDependencyFilter

```
public net.vpc.app.nuts.NutsDependencyFilter
```

Dependency filter

⚙️ Instance Methods

⚙️ accept(from, dependency, session)

return true if the `dependency` is accepted

```
boolean accept(NutsId from, NutsDependency dependency, NutsSession session)
```

parent (dependent) id dependency id session

☕ NutsDependencyTreeNode

```
public net.vpc.app.nuts.NutsDependencyTreeNode
```

Dependency tree node

Instance Properties

children

node children

```
[read-only] NutsDependencyTreeNode[] children  
NutsDependencyTreeNode[] getChildren()
```

dependency

node dependency

```
[read-only] NutsDependency dependency  
NutsDependency getDependency()
```

partial

true if the node is partial filled (not all children are considered)

```
[read-only] boolean partial  
boolean isPartial()
```

NutsDescriptor

```
public net.vpc.app.nuts.NutsDescriptor
```

Nuts descriptors define an immutable image to all information needed to execute an artifact. It resembles to maven's pom file but it focuses on execution information rather than build information. Common features are inheritance dependencies, standard dependencies, exclusions and properties. However nuts descriptor adds new features such as :

- multiple parent inheritance
- executable/nuts-executable flag

- environment (arch, os, dist, platform) filters
- classifiers may be mapped to environment (think of dlls for windows and so for linux)

A versatile way to change descriptor is to use builder (`#builder()`). ===== Instance Properties

application

true if the artifact is a java executable that implements ` NutsApplication` interface.

```
[read-only] boolean application
boolean isApplication()
```

arch

supported archs. if empty, all arch are supported (for example for java, all arch are supported).

```
[read-only] String[] arch
String[] getArch()
```

classifierMappings

ordered list of classifier mapping used to resolve valid classifier to use of a given environment.

```
[read-only] NutsClassifierMapping[] classifierMappings
NutsClassifierMapping[] getClassifierMappings()
```

dependencies

list of immediate (non inherited and non transitive) dependencies

```
[read-only] NutsDependency[] dependencies
NutsDependency[] getDependencies()
```

description

long description for the artifact

```
[read-only] String description  
String getDescription()
```

executable

true if the artifact is executable and is considered an application. if not it is a library.

```
[read-only] boolean executable  
boolean isExecutable()
```

executor

descriptor of artifact responsible of running this artifact

```
[read-only] NutsArtifactCall executor  
NutsArtifactCall getExecutor()
```

id

artifact full id (groupId+artifactId+version)

```
[read-only] NutsId id  
NutsId getId()
```

installer

descriptor of artifact responsible of installing this artifact

```
[read-only] NutsArtifactCall installer  
NutsArtifactCall getInstaller()
```

locations

list of available mirror locations from which nuts can download artifact content. location can be mapped to a classifier.

```
[read-only] NutsIdLocation[] locations  
NutsIdLocation[] getLocations()
```

name

user friendly name, a short description for the artifact

```
[read-only] String name  
String getName()
```

os

supported operating systems. if empty, all oses are supported (for example for java, all arch are supported).

```
[read-only] String[] os  
String[] getOs()
```

osdist

supported operating system distributions (mostly for linux systems). if empty, all distributions are supported.

```
[read-only] String[] osdist  
String[] getOsdist()
```

packaging

return descriptor packaging (used to resolve file extension)

```
[read-only] String packaging  
String getPackaging()
```

📄 parents

descriptor parent list (may be empty)

```
[read-only] NutsId[] parents  
NutsId[] getParents()
```

📄 platform

supported platforms (java, dotnet, ...). if empty platform is not relevant. This is helpful to bind application to a jdk version for instance (in that case platform may be in the form java#8 for instance)

```
[read-only] String[] platform  
String[] getPlatform()
```

📄 properties

custom properties that can be used as place holders (int form) in other fields.

```
[read-only] Map<String, String> properties  
Map<String, String> getProperties()
```

📄 standardDependencies

The dependencies specified here are not used until they are referenced in a POM within the group. This allows the specification of a "standard" version for a particular. This corresponds to "dependencyManagement.dependencies" in maven

```
[read-only] NutsDependency[] standardDependencies  
NutsDependency[] getStandardDependencies()
```

⚙️ Instance Methods

⚙️ builder()

create new builder filled with this descriptor fields.

```
NutsDescriptorBuilder builder()
```

✍️ NutsDescriptorBuilder

```
public net.vpc.app.nuts.NutsDescriptorBuilder
```

Nuts descriptors define a mutable image to all information needed to execute an artifact. It helps creating an instance of `NutsDescriptor` by calling `#build()`
==== Instance Properties

㉒ application

set nutsApp flag

```
[write-only] NutsDescriptorBuilder application  
NutsDescriptorBuilder setApplication(nutsApp)
```

📝 arch

set archs

```
[read-write] NutsDescriptorBuilder arch  
String[] getArch()  
NutsDescriptorBuilder setArch(archs)
```

📝 classifierMappings

set classifier mappings

```
[read-write] NutsDescriptorBuilder classifierMappings  
NutsClassifierMapping[] getClassifierMappings()  
NutsDescriptorBuilder setClassifierMappings(value)
```

dependencies

set dependencies

```
[read-write] NutsDescriptorBuilder dependencies  
NutsDependency[] getDependencies()  
NutsDescriptorBuilder setDependencies(dependencies)
```

description

set description

```
[read-write] NutsDescriptorBuilder description  
String getDescription()  
NutsDescriptorBuilder setDescription(description)
```

executable

set executable flag

```
[read-write] NutsDescriptorBuilder executable  
boolean isExecutable()  
NutsDescriptorBuilder setExecutable(executable)
```

executor

set executor flag

```
[read-write] NutsDescriptorBuilder executor  
NutsArtifactCall getExecutor()  
NutsDescriptorBuilder setExecutor(executor)
```

id

set id

```
[read-write] NutsDescriptorBuilder id  
NutsId getId()  
NutsDescriptorBuilder setId(id)
```

installer

set installer

```
[read-write] NutsDescriptorBuilder installer  
NutsArtifactCall getInstaller()  
NutsDescriptorBuilder setInstaller(installer)
```

locations

set locations

```
[read-write] NutsDescriptorBuilder locations  
NutsIdLocation[] getLocations()  
NutsDescriptorBuilder setLocations(locations)
```

name

set name

```
[read-write] NutsDescriptorBuilder name  
String getName()  
NutsDescriptorBuilder setName(name)
```

nutsApplication

true if the artifact is a java executable that implements `NutsApplication` interface.

```
[read-only] boolean nutsApplication  
boolean isNutsApplication()
```

os

set os

```
[read-write] NutsDescriptorBuilder os
String[] getOs()
NutsDescriptorBuilder setOs(os)
```

osdist

set osdist

```
[read-write] NutsDescriptorBuilder osdist
String[] getOsdist()
NutsDescriptorBuilder setOsdist(osdist)
```

packaging

set packaging

```
[read-write] NutsDescriptorBuilder packaging
String getPackaging()
NutsDescriptorBuilder setPackaging(packaging)
```

parents

set parents

```
[read-write] NutsDescriptorBuilder parents
NutsId[] getParents()
NutsDescriptorBuilder setParents(parents)
```

platform

set platform

```
[read-write] NutsDescriptorBuilder platform  
String[] getPlatform()  
NutsDescriptorBuilder setPlatform(platform)
```

properties

set properties

```
[read-write] NutsDescriptorBuilder properties  
Map<String, String> getProperties()  
NutsDescriptorBuilder setProperties(properties)
```

standardDependencies

set standard dependencies

```
[read-write] NutsDescriptorBuilder standardDependencies  
NutsDependency[] getStandardDependencies()  
NutsDescriptorBuilder setStandardDependencies(dependencies)
```

Instance Methods

addArch(arch)

add arch

```
NutsDescriptorBuilder addArch(String arch)
```

new value to add

addClassifierMapping(mapping)

add classifier mapping

```
NutsDescriptorBuilder addClassifierMapping(NutsClassifierMapping mapping)
```

classifier mapping

⚙️ addDependencies(dependencies)

add dependencies

```
NutsDescriptorBuilder addDependencies(NutsDependency[] dependencies)
```

new value to add

⚙️ addDependency(dependency)

add dependency

```
NutsDescriptorBuilder addDependency(NutsDependency dependency)
```

new value to add

⚙️ addLocation(location)

add location

```
NutsDescriptorBuilder addLocation(NutsIdLocation location)
```

location to add

⚙️ addOs(os)

add os

```
NutsDescriptorBuilder addOs(String os)
```

new value to add

⚙️ addOsdist(osdist)

add os dist

```
NutsDescriptorBuilder addOsdist(String osdist)
```

new value to add

⚙️ `addPlatform(platform)`

add platform

```
NutsDescriptorBuilder addPlatform(String platform)
```

new value to add

⚙️ `addProperties(properties)`

merge properties

```
NutsDescriptorBuilder addProperties(Map<String, String> properties)
```

new value

⚙️ `addStandardDependencies(dependencies)`

add standard dependencies

```
NutsDescriptorBuilder addStandardDependencies(NutsDependency[] dependencies)
```

value to add

⚙️ `addStandardDependency(dependency)`

add standard dependency

```
NutsDescriptorBuilder addStandardDependency(NutsDependency dependency)
```

value to add

⚙️ **application()**

```
NutsDescriptorBuilder application()
```

⚙️ **application(nutsApp)**

```
NutsDescriptorBuilder application(boolean nutsApp)
```

null

⚙️ **applyParents(parentDescriptors)**

merge parent and child information (apply inheritance)

```
NutsDescriptorBuilder applyParents(NutsDescriptor[] parentDescriptors)
```

parent descriptors

⚙️ **applyProperties()**

replace placeholders with the corresponding property value in properties list

```
NutsDescriptorBuilder applyProperties()
```

⚙️ **applyProperties(properties)**

replace placeholders with the corresponding property value in the given properties list and return a new instance.

```
NutsDescriptorBuilder applyProperties(Map<String, String> properties)
```

properties

⚙ arch(archs)

set archs

```
NutsDescriptorBuilder arch(String[] archs)
```

value to set

⚙ build()

create new Descriptor filled with this builder fields.

```
NutsDescriptor build()
```

⚙ classifierMappings(value)

```
NutsDescriptorBuilder classifierMappings(NutsClassifierMapping[] value)
```

null

⚙ clear()

clear this instance (set null/default all properties)

```
NutsDescriptorBuilder clear()
```

⚙ dependencies(dependencies)

set dependencies

```
NutsDescriptorBuilder dependencies(NutsDependency[] dependencies)
```

new value

⚙️ **description(description)**

```
NutsDescriptorBuilder description(String description)
```

null

⚙️ **descriptor(other)**

```
NutsDescriptorBuilder descriptor(NutsDescriptor other)
```

null

⚙️ **descriptor(other)**

```
NutsDescriptorBuilder descriptor(NutsDescriptorBuilder other)
```

null

⚙️ **executable()**

```
NutsDescriptorBuilder executable()
```

⚙️ **executable(executable)**

```
NutsDescriptorBuilder executable(boolean executable)
```

null

⚙️ **executor(executor)**

```
NutsDescriptorBuilder executor(NutsArtifactCall executor)
```

null

⚙️ id(id)

```
NutsDescriptorBuilder id(NutsId id)
```

null

⚙️ installer(installer)

```
NutsDescriptorBuilder installer(NutsArtifactCall installer)
```

null

⚙️ locations(locations)

```
NutsDescriptorBuilder locations(NutsIdLocation[] locations)
```

null

⚙️ name(name)

set name

```
NutsDescriptorBuilder name(String name)
```

value to set

⚙️ os(os)

set os

```
NutsDescriptorBuilder os(String[] os)
```

value to set

⚙️ **osdist(osdist)**

set osdist

NutsDescriptorBuilder **osdist**(String[] osdist)

value to set

⚙️ **packaging(packaging)**

set packaging

NutsDescriptorBuilder **packaging**(String packaging)

new value

⚙️ **parents(parents)**

set parents

NutsDescriptorBuilder **parents**(NutsId[] parents)

value to set

⚙️ **platform(platform)**

set platform

NutsDescriptorBuilder **platform**(String[] platform)

value to set

⚙️ **properties(properties)**

set properties

```
NutsDescriptorBuilder properties(Map<String, String> properties)
```

new value

⚙️ `property(name, value)`

```
NutsDescriptorBuilder property(String name, String value)
```

null null

⚙️ `removeArch(arch)`

remove arch

```
NutsDescriptorBuilder removeArch(String arch)
```

value to remove

⚙️ `removeDependency(dependency)`

remove dependency

```
NutsDescriptorBuilder removeDependency(NutsDependency dependency)
```

value to remove

⚙️ `removeDependency(dependency)`

create a new instance of descriptor with removed dependencies that match the predicate

```
NutsDescriptorBuilder removeDependency(Predicate<NutsDependency> dependency)
```

predicate to test against

 **removeOs(os)**

remove os

NutsDescriptorBuilder `removeOs(String os)`

value to remove

 **removeOsdist(osdist)**

remove osdist

NutsDescriptorBuilder `removeOsdist(String osdist)`

value to remove

 **removePlatform(platform)**

remove platform

NutsDescriptorBuilder `removePlatform(String platform)`

value to remove

 **removeStandardDependency(dependency)**

remove standard dependency

NutsDescriptorBuilder `removeStandardDependency(NutsDependency dependency)`

value to remove

 **replaceDependency(filter, converter)**

create a new instance of descriptor with added/merged dependencies

```
NutsDescriptorBuilder replaceDependency(Predicate<NutsDependency> filter,  
UnaryOperator<NutsDependency> converter)
```

properties entry that match the update function to provide new value to replace with

⚙️ replaceProperty(filter, converter)

create a new instance of descriptor with added/merged properties

```
NutsDescriptorBuilder replaceProperty(Predicate<Map.Entry<String, String>>  
filter, Function<Map.Entry<String, String>, String> converter)
```

properties entry that match the update function to provide new value to replace with

⚙️ set(other)

set all fields from `other`

```
NutsDescriptorBuilder set(NutsDescriptorBuilder other)
```

builder to copy from

⚙️ set(other)

set all fields from `other`

```
NutsDescriptorBuilder set(NutsDescriptor other)
```

descriptor to copy from

⚙️ setProperty(name, value)

set or unset property. if the value is null, the property is removed.

```
NutsDescriptorBuilder setProperty(String name, String value)
```

property name new value

⚙️ **standardDependencies(dependencies)**

set standard dependencies

```
NutsDescriptorBuilder standardDependencies(NutsDependency[] dependencies)
```

value to set

☕️ **NutsDescriptorFilter**

```
public net.vpc.app.nuts.NutsDescriptorFilter
```

Descriptor filter

⚙️ **Instance Methods**

⚙️ **accept(descriptor, session)**

return true if descriptor is accepted

```
boolean accept(NutsDescriptor descriptor, NutsSession session)
```

descriptor session

⚙️ **acceptSearchId(sid, session)**

default implementation of `NutsSearchIdFilter`

```
boolean acceptSearchId(NutsSearchId sid, NutsSession session)
```

search id session

☕️ **NutsExecutableInformation**

```
public net.vpc.app.nuts.NutsExecutableInformation
```

Class describing executable command.

Instance Properties

description

executable description

```
[read-only] String description  
String getDescription()
```

helpText

executable help string

```
[read-only] String helpText  
String getHelpText()
```

id

executable artifact id

```
[read-only] NutsId id  
NutsId getId()
```

name

executable name

```
[read-only] String name  
String getName()
```

📄 type

return executable type

```
[read-only] NutsExecutableType type  
NutsExecutableType getType()
```

📄 value

versatile executable name

```
[read-only] String value  
String getValue()
```

☕ NutsExecutionEntry

```
public net.vpc.app.nuts.NutsExecutionEntry
```

Execution entry is a class that can be executed.

🔗 Instance Properties

📄 ↴ app

true if the entry resolved to a valid nuts application

```
[read-only] boolean app  
boolean isApp()
```

📄 ↴ defaultEntry

true if the class if registered as main class in META-INF

```
[read-only] boolean defaultEntry  
boolean isDefaultEntry()
```

 [name](#)

class name

```
[read-only] String name  
String getName()
```

 [NutsId](#)

```
public net.vpc.app.nuts.NutsId
```

Immutable Artifact id information.

[Instance Properties](#) [arch](#)

hardware architecture supported by the artifact

```
[read-only] String arch  
String getArch()
```

 [artifactId](#)

return name part of this id

```
[read-only] String artifactId  
String getArtifactId()
```

 [classifier](#)

tag used to distinguish between different artifacts that were built from the same source code

```
[read-only] String classifier  
String getClassifier()
```

 **face**

id face define is a release file type selector of the id. It helps discriminating content (jar) from descriptor, from other (hash,...) files released for the very same artifact.

```
[read-only] String face  
String getFace()
```

 **fullName**

return a string representation of this id. All of group, name, version, namespace, queryMap values are printed. This method is equivalent to `Object#toString()`

```
[read-only] String fullName  
String getFullName()
```

 **groupId**

artifact group which identifies uniquely projects and group of projects.

```
[read-only] String groupId  
String getGroupId()
```

 **longName**

return a string concatenation of group, name and version, ignoring namespace, and queryMap values. An example of long name is

```
my-group:my-artifact#my-version?alt
```

```
[read-only] String longName  
String getLongName()
```

📄 longNameId

return a new instance of NutsId defining only group, name and version, ignoring namespace, and queryMap values.

```
[read-only] NutsId longNameId  
NutsId getLongNameId()
```

📄 namespace

artifact namespace (usually repository name or id)

```
[read-only] String namespace  
String getNamespace()
```

📄 os

os supported by the artifact

```
[read-only] String os  
String getOs()
```

📄 osdist

os distribution supported by the artifact

```
[read-only] String osdist  
String getOsdist()
```

📄 platform

platform supported by the artifact

```
[read-only] String platform  
String getPlatform()
```

properties

properties as map.

```
[read-only] Map<String, String> properties
Map<String, String> getProperties()
```

propertiesQuery

properties in the url query form

```
[read-only] String propertiesQuery
String getPropertiesQuery()
```

shortName

returns a string concatenation of group and name (':' separated) ignoring version,namespace, and queryMap values. In group is empty or null, name is returned. Ann null values are trimmed to "" An example of simple name is `` my-group:my-artifact

```
[read-only] String shortName String getShortName()
```

```
####  shortNameId
return a new instance of NutsId defining only group and name ignoring
version,namespace, and queryMap values.
```

```
[read-only] NutsId shortNameId NutsId getShortNameId()
```

```
####  version
artifact version (never null)
```

```
[read-only] NutsVersion version NutsVersion getVersion()
```

```
### ⚙ Instance Methods  
#### ⚙ anyToken()  
non null token filter that searches in all id fields
```

NutsTokenFilter anyToken()

```
#### ⚙ artifactIdToken()  
non null artifact id token
```

NutsTokenFilter artifactIdToken()

```
#### ⚙ builder()  
create a builder (mutable id) based on this id
```

NutsIdBuilder builder()

```
#### ⚙ equalsShortName(other)  
true if other has exact short name than `'' this````
```

boolean equalsShortName(NutsId other)

```
other id  
#### ⚙ filter()  
create a filter based on this id
```

NutsIdFilter filter()

```
#### ⚙ groupIdToken()  
non null group id token
```

NutsTokenFilter groupIdToken()

```
#### ⚙ namespaceToken()  
non null namespace non null namespace token
```

NutsTokenFilter namespaceToken()

```
#### 🔒 propertiesToken()  
non null properties query token
```

NutsTokenFilter propertiesToken()

```
#### 🔒 versionToken()  
non null version token
```

NutsTokenFilter versionToken()

```
## 📥 NutsIdBuilder
```

public net.vpc.app.nuts.NutsIdBuilder

```
Mutable Artifact id information used to create instance of ``` NutsId```\n### 🚦 Instance Properties\n#### 🖊 arch  
update arch
```

[read-write] NutsIdBuilder arch String getArch() NutsIdBuilder setArch(value)

```
#### 🖊 artifactId  
update artifactId
```

[read-write] NutsIdBuilder artifactId String getArtifactId() NutsIdBuilder setArtifactId(value)

```
#### 🖊 classifier  
update classifier
```

[read-write] NutsIdBuilder classifier String getClassifier() NutsIdBuilder setClassifier(value)

```
#### 🖊 face  
update id face which defines is a release file type selector
```

[read-write] NutsIdBuilder face String getFace() NutsIdBuilder setFace(value)

fullName

return a string representation of this id. All of group, name, version, namespace, queryMap values are printed. This method is equivalent to

[read-only] String **fullName**
String **getFullName()**

groupId

update groupId

[read-write] NutsIdBuilder groupId
String **getGroupId()**
NutsIdBuilder **setGroupId**(value)

longName

return a string concatenation of group, name and version, ignoring namespace, and queryMap values. An example of long name is

my-group:my-artifact#my-version?alt

[read-only] String longName
String **getLongName()**

namespace

update namespace

[read-write] NutsIdBuilder namespace
String **getNamespace()**
NutsIdBuilder **setNamespace**(value)

os

update os

```
[read-write] NutsIdBuilder os
String getOs()
NutsIdBuilder setOs(value)
```

osdist

update osdist

```
[read-write] NutsIdBuilder osdist
String getOsdist()
NutsIdBuilder setOsdist(value)
```

packaging

update packaging

```
[write-only] NutsIdBuilder packaging
NutsIdBuilder setPackaging(packaging)
```

platform

update platform

```
[read-write] NutsIdBuilder platform
String getPlatform()
NutsIdBuilder setPlatform(value)
```

properties

update all properties property.

```
[read-write] NutsIdBuilder properties
Map<String, String> getProperties()
NutsIdBuilder setProperties(query)
```

propertiesQuery

properties in the url query form

```
[read-only] String propertiesQuery
String getPropertiesQuery()
```

shortName

returns a string concatenation of group and name (':' separated) ignoring version,namespace, and queryMap values. In group is empty or null, name is returned. Ann null values are trimmed to "" An example of simple name is `` my-group:my-artifact

```
[read-only] String shortName String getShortName()
```

```
####  version
update setVersion
```

```
[read-write] NutsIdBuilder version NutsVersion getVersion() NutsIdBuilder setVersion(value)
```

```
###  Instance Methods
####  addProperties(query)
update all properties property while retaining old,
non overridden properties.
```

NutsIdBuilder addProperties(String query)

new value

⚙ addProperties(Map<String, String> queryMap)
update all properties property while retaining old,
non overridden properties.

NutsIdBuilder addProperties(Map<String, String> queryMap)

new value

⚙ apply(Function<String, String> properties)
replace dollar based variables with the given properties

NutsIdBuilder apply(Function<String, String> properties)

to replace

⚙ build()
create new instance of `` `NutsId``` initialized with this builder values.

NutsId build()

⚙ clear()
clear this instance (set null/default all properties)

NutsIdBuilder clear()

⚙ set(id)
update all arguments

NutsIdBuilder set(NutsId id)

new value

⚙ set(id)
update all arguments

NutsIdBuilder set(NutsIdBuilder id)

new value

⚙️ setFaceContent()
equivalent to ```setFace(NutsConstants.QueryFaces.CONTENT)```

NutsIdBuilder setFaceContent()

⚙️ setFaceDescriptor()
equivalent to ```setFace(NutsConstants.QueryFaces.DESCRIPTOR)```

NutsIdBuilder setFaceDescriptor()

⚙️ setProperty(property, value)
update property.
When ```value``` is null, property will be removed.

NutsIdBuilder setProperty(String property, String value)

name
new value

☕ NutsIdFilter

public net.vpc.app.nuts.NutsIdFilter

Class for filtering Artifact Ids
⚙️ Instance Methods
⚙️ accept(id, session)
return true when the id is to be accepted

boolean accept(NutsId id, NutsSession session)

id to check
current workspace session

☕ acceptSearchId(sid, session)

boolean acceptSearchId(NutsSearchId sid, NutsSession session)

null
null

☕ NutsIdLocation

public net.vpc.app.nuts.NutsIdLocation

This class is used in ``` NutsDescriptor``` to describe
locations/mirrors to download artifact content instead of the
regular location.

□ Instance Properties
📄□ classifier
classifier for the artifact

[read-only] String classifier String getClassifier()

📄□ region
location (geographic) region that may be used to select
the most effective mirror

[read-only] String region String getRegion()

📄□ url
location url of the artifact content

[read-only] String url String getUrl()

☕ NutsVersion

public net.vpc.app.nuts.NutsVersion

this class represents an ****immutable**** string representation of a version parsed as a suite of alternating numbers and words.

Parsing algorithm is simply to split whenever word type changes.

Examples:

- 1 = [1]
- 1.2 = [1,'.',2]
- 10.20update3 = [10,'.',20,'update',3]

Instance Properties

**singleValue**

return true if this version denotes as single value and does not match an interval.

[read-only] boolean singleValue boolean isSingleValue()

**value**

return string representation of the version

[read-only] String value String getValue()

Instance Methods

**compareTo(other)**

compare this version to the other version

int compareTo(String other)

other version

**compareTo(other)**

int compareTo(NutsVersion other)

null

**filter()**

parse the current version as new instance of ```` NutsVersionFilter````

NutsVersionFilter filter()

```
#### ⚙ get(index)
element at given index. if the index is negative will return from right.
```

- (1.a22).get(0)=1
- (1.a22).get(1)=a
- (1.a22).get(-1)=22

String get(int index)

version part index

```
#### ⚙ getNumber(index)
number element at given index. if the index is negative will return from right (-1 is the first starting from the right).
```

The version is first split (as a suite of number and words) then all words are discarded.

- size(1.22)=3 {'1','.','22'}
- size(1.22_u1)=5 {'1','.','22','_u','1'}

- (1.a22).getNumber(0)=1
- (1.a22).getNumber(1)=22
- (1.a22).getNumber(-1)=22

int getNumber(int index)

version part index

```
#### ⚙ getNumber(index, defaultValue)
```

int getNumber(int index, int defaultValue)

null
null

```
#### ⚙ inc()
increment the last number in the version with 1
```

NutsVersion inc()

```
#### ⚙ inc(position)
increment the number at ```position``` in the version with 1
```

NutsVersion inc(int position)

number position

```
#### ⚙ inc(position, amount)
increment the last number in the version with the given ```amount```
```

NutsVersion inc(int position, int amount)

number position
amount of the increment

```
#### ⚙ intervals()
parse the current version as an interval array
```

NutsVersionInterval[] intervals()

```
#### ⚙ numberSize()
number of elements in the version.
```

- numberSize(1.22)=2 {1,22}
- numberSize(1.22_u1)=3 {1,22,1}

int numberSize()

```
#### ⚙ size()
number of elements in the version.
```

- size(1.22)=3 {'1','.','22'}
- size(1.22_u1)=5 {'1','.','22','_u','1'}

int size()

```
## ☕ NutsVersionFilter
```

public net.vpc.app.nuts.NutsVersionFilter

version interval is a version filter that accepts interval ranges of versions.

version intervals can be in one of the following forms

[version, [] version,] or (version,] [version, [or [version,)[][] version, [or [version, [[[,version [] ,version] or (,version] [,version [or [,version)[][] ,version [or [,version [[[version1 , version2 [] version1 , version2] or (version1 , version2] [version1 , version2 [or [version1 , version2)[][] version1 , version2 [or [version1 , version2 [[comma or space separated intervals such as : [version1 , version2 [. [version1 , version2 [[version1 , version2 [[version1 , version2 [

Created by vpc on 1/8/17.

⚙ Instance Methods

⚙ accept(version, session)

true if the version is accepted by this instance filter

boolean accept(NutsVersion version, NutsSession session)

version to check

current session instance

⚙ acceptSearchId(sid, session)

true if the version is accepted by this instance filter

boolean acceptSearchId(NutsSearchId sid, NutsSession session)

search id

current session instance

Events

☕ NutsInstallListener

```
public net.vpc.app.nuts.NutsInstallListener
```

A class can implement the `` NutsInstallListener

wants to be informed of install artifacts actions.

⚙ Instance Methods

⚙ onInstall(event)

This method is called whenever the observed workspace installs an artifact.

```
void onInstall(NutsInstallEvent event)
```

event

⚙ onUninstall(event)

This method is called whenever the observed workspace uninstalls an artifact.

```
void onUninstall(NutsInstallEvent event)
```

event

⚙ onUpdate(event)

This method is called whenever the observed workspace updates an artifact.

```
void onUpdate(NutsUpdateEvent event)
```

event

☕ NutsRepositoryEvent

```
public net.vpc.app.nuts.NutsRepositoryEvent
```

Repository Event

Instance Properties

parent

Parent repository when this event is about creating
a new repository with a parent one.

[read-only[NutsRepository parent NutsRepository getParent()]

propertyName

event property name

[read-only[String propertyName String getPropertyName()]

propertyOldValue

event property old value

[read-only[Object propertyOldValue Object getPropertyOldValue()]

PropertyValue

event property new value

[read-only[Object PropertyValue Object getPropertyValue()]

repository

repository that fires this event or the new repository
when creating a new one with parent.

[read-only[NutsRepository repository NutsRepository getRepository()]

session

current session

[read-only[NutsSession session NutsSession getSession()]

workspace

current workspace

[read-only] NutsWorkspace workspace NutsWorkspace getWorkspace()

```
## ☕ NutsRepositoryListener
```

public net.vpc.app.nuts.NutsRepositoryListener

Created by vpc on 1/20/17.

⚙ Instance Methods

```
#### ⚙ onAddRepository(event)
```

void onAddRepository(NutsRepositoryEvent event)

null

```
#### ⚙ onConfigurationChanged(event)
```

void onConfigurationChanged(NutsRepositoryEvent event)

null

```
#### ⚙ onDeploy(event)
```

void onDeploy(NutsContentEvent event)

null

```
#### ⚙ onPush(event)
```

void onPush(NutsContentEvent event)

null

```
#### ⚙ onRemoveRepository(event)
```

void onRemoveRepository(NutsRepositoryEvent event)

```
null
```

```
#### ⚙️ onUndeploy(event)
```

```
void onUndeploy(NutsContentEvent event)
```

```
null
```

```
## ☕ NutsUpdateEvent
```

```
public net.vpc.app.nuts.NutsUpdateEvent
```

```
### 📄 Instance Properties
```

```
#### 📋 force
```

```
[read-only[ boolean force boolean isForce()
```

```
#### 📋 newValue
```

```
[read-only[ NutsDefinition newValue NutsDefinition getnewValue()
```

```
#### 📋 oldValue
```

```
[read-only[ NutsDefinition oldValue NutsDefinition getoldValue()
```

```
#### 📋 session
```

```
[read-only[ NutsSession session NutsSession getSession()
```

```
#### 📋 workspace
```

```
[read-only[ NutsWorkspace workspace NutsWorkspace getWorkspace()
```

```
## ☕ NutsWorkspaceEvent
```

```
public net.vpc.app.nuts.NutsWorkspaceEvent
```

```
### □ Instance Properties
```

```
#### 📄 □ propertyName
```

```
[read-only] String propertyName String getPropertyName()
```

```
#### 📄 □ propertyOldValue
```

```
[read-only] Object propertyOldValue Object getPropertyOldValue()
```

```
#### 📄 □ PropertyValue
```

```
[read-only] Object PropertyValue Object getPropertyValue()
```

```
#### 📄 □ repository
```

```
[read-only] NutsRepository repository NutsRepository getRepository()
```

```
#### 📄 □ session
```

```
[read-only] NutsSession session NutsSession getSession()
```

```
#### 📄 □ workspace
```

```
[read-only] NutsWorkspace workspace NutsWorkspace getWorkspace()
```

```
## ☕ NutsWorkspaceListener
```

```
public net.vpc.app.nuts.NutsWorkspaceListener
```

```
Created by vpc on 1/20/17.
```

```
### ⚙ Instance Methods
```

```
#### ⚙ onAddRepository(event)
```

```
void onAddRepository(NutsWorkspaceEvent event)
```

```
null
```

```
#### ⚙ onConfigurationChanged(event)
```

```
void onConfigurationChanged(NutsWorkspaceEvent event)
```

```
null
```

```
#### ⚙ onCreateWorkspace(event)
```

```
void onCreateWorkspace(NutsWorkspaceEvent event)
```

```
null
```

```
#### ⚙ onReloadWorkspace(event)
```

```
void onReloadWorkspace(NutsWorkspaceEvent event)
```

```
null
```

```
#### ⚙ onRemoveRepository(event)
```

```
void onRemoveRepository(NutsWorkspaceEvent event)
```

```
null
```

```
#### ⚙ onUpdateProperty(event)
```

```
void onUpdateProperty(NutsWorkspaceEvent event)
```

```
null
```

Exception

⌚ NutsAlreadyDeployedException

```
public net.vpc.app.nuts.NutsAlreadyDeployedException
```

Exception fired in `NutsWorkspace#deploy()` method if the package is already deployed Created by vpc on 1/15/17.

Constructors

⌚ NutsAlreadyDeployedException(workspace, id)

Custom Constructor

```
NutsAlreadyDeployedException(NutsWorkspace workspace, NutsId id)
```

workspace nuts id

⌚ NutsAlreadyDeployedException(workspace, id)

Custom Constructor

```
NutsAlreadyDeployedException(NutsWorkspace workspace, String id)
```

workspace nuts id

⌚ NutsAlreadyDeployedException(workspace, id, msg, cause)

Custom Constructor

```
NutsAlreadyDeployedException(NutsWorkspace workspace, String id, String msg,  
Exception cause)
```

workspace nuts id message cause

[NutsAlreadyDeployedException\(workspace, id, msg, ex\)](#)

Custom Constructor

```
NutsAlreadyDeployedException(NutsWorkspace workspace, NutsId id, String msg,  
Exception ex)
```

workspace nuts id message exception

[NutsAlreadyInstalledException](#)

```
public net.vpc.app.nuts.NutsAlreadyInstalledException
```

Thrown to indicate that the artifact is already installed and should not be reinstalled Created by vpc on 1/15/17.

[Constructors](#)

[NutsAlreadyInstalledException\(workspace, id\)](#)

Custom Constructor

```
NutsAlreadyInstalledException(NutsWorkspace workspace, NutsId id)
```

workspace nuts id

[NutsAlreadyInstalledException\(workspace, id\)](#)

Custom Constructor

```
NutsAlreadyInstalledException(NutsWorkspace workspace, String id)
```

workspace nuts id

[NutsAlreadyInstalledException\(workspace, id, message, cause\)](#)

Custom Constructor

```
NutsAlreadyInstalledException(NutsWorkspace workspace, NutsId id, String message,
Exception cause)
```

workspace nuts id message exception

[? NutsAlreadyInstalledException\(workspace, id, msg, cause\)](#)

Custom Constructor

```
NutsAlreadyInstalledException(NutsWorkspace workspace, String id, String msg,
Exception cause)
```

workspace nuts id message exception

 [NutsElementException](#)

```
public net.vpc.app.nuts.NutsElementException
```

Generic exception to be thrown when an element is not found.

[? Constructors](#)

[? NutsElementException\(workspace\)](#)

Constructs a new runtime exception with `null` as itsdetail message. The cause is not initialized, and may subsequently be initialized by a call to `#initCause` .

```
NutsElementException(NutsWorkspace workspace)
```

the workspace of this Nuts Exception

[? NutsElementException\(workspace, cause\)](#)

Constructs a new runtime exception with the specified cause and a detail message of (**cause==null** ? **null** : **cause.toString()**) (which typically contains the class and detail message of **cause**). This constructor is useful for runtime exceptionsthat are little more than wrappers for other

throwables.

```
NutsElementException(NutsWorkspace workspace, Throwable cause)
```

the workspace of this Nuts Exception the cause (which is saved for later retrieval by the {@link #getCause()} method). (A <tt>null</tt> value is permitted, and indicates that the cause is nonexistent or unknown.)

[¶ NutsElementException\(workspace, message\)](#)

Constructs a new runtime exception with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to `#initCause` .

```
NutsElementException(NutsWorkspace workspace, String message)
```

the workspace of this Nuts Exception the detail message. The detail message is saved for later retrieval by the {@link #getMessage()} method.

[¶ NutsElementException\(workspace, message, cause\)](#)

Constructs a new runtime exception with the specified detail message and cause.

Note that the detail message associated with
`cause` is not automatically incorporated in this runtime exception's detail message.

```
NutsElementException(NutsWorkspace workspace, String message, Throwable cause)
```

the workspace of this Nuts Exception the detail message (which is saved for later retrieval by the {@link #getMessage()} method). the cause (which is saved for later retrieval by the {@link #getCause()} method). (A <tt>null</tt> value is permitted, and indicates that the cause is nonexistent or unknown.)

[? NutsElementException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new runtime exception with the specified detail message, cause, suppression enabled or disabled, and writable stack trace enabled or disabled.

```
NutsElementException(NutsWorkspace workspace, String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace)
```

the workspace of this Nuts Exception the detail message. the cause. (A {@code null} value is permitted, and indicates that the cause is nonexistent or unknown.) whether or not suppression is enabled or disabled whether or not the stack trace should be writable

 [NutsException](#)

```
public net.vpc.app.nuts.NutsException
```

Base Nuts Exception. Parent of all Nuts defined Exceptions.

[? Constructors](#)

[? NutsException\(workspace\)](#)

Constructs a new runtime exception with `null` as itsdetail message. The cause is not initialized, and may subsequently be initialized by a call to `#initCause` .

```
NutsException(NutsWorkspace workspace)
```

the workspace of this Nuts Exception

[? NutsException\(workspace, cause\)](#)

Constructs a new runtime exception with the specified cause and a detail message of (`cause==null ? null : cause.toString()`) (which typically contains the class and detail message of `cause`). This constructor is useful for runtime exceptionsthat are little more than wrappers for other throwables.

NutsException(NutsWorkspace workspace, Throwable cause)

the workspace of this Nuts Exception the cause (which is saved for later retrieval by the {@link #getCause()} method). (A <tt>null</tt> value is permitted, and indicates that the cause is nonexistent or unknown.)

[NutsException\(workspace, cause\)](#)

Constructs a new runtime exception with the specified cause and a detail message of (`cause==null ? null : cause.toString()`) (which typically contains the class and detail message of `cause`). This constructor is useful for runtime exceptionsthat are little more than wrappers for other throwables.

NutsException(NutsWorkspace workspace, IOException cause)

the workspace of this Nuts Exception the cause (which is saved for later retrieval by the {@link #getCause()} method). (A <tt>null</tt> value is permitted, and indicates that the cause is nonexistent or unknown.)

[NutsException\(workspace, message\)](#)

Constructs a new runtime exception with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to `#initCause` .

NutsException(NutsWorkspace workspace, String message)

the workspace of this Nuts Exception the detail message. The detail message is saved for later retrieval by the {@link #getMessage()} method.

[NutsException\(workspace, message, cause\)](#)

Constructs a new runtime exception with the specified detail message and cause.

Note that the detail message associated with `cause` is not automatically incorporated inthis runtime exception's detail message.

`NutsException(NutsWorkspace workspace, String message, Throwable cause)`

the workspace of this Nuts Exception the detail message (which is saved for later retrieval by the {@link #getMessage()} method). the cause (which is saved for later retrieval by the {@link #getCause()} method). (A <tt>null</tt> value is permitted, and indicates that the cause is nonexistent or unknown.)

[? NutsException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new runtime exception with the specified detail message, cause, suppression enabled or disabled, and writable stack trace enabled or disabled.

`NutsException(NutsWorkspace workspace, String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace)`

the workspace of this Nuts Exception the detail message. the cause. (A {@code null} value is permitted, and indicates that the cause is nonexistent or unknown.) whether or not suppression is enabled or disabled whether or not the stack trace should be writable

[? Instance Properties](#)

 [? workspace](#)

Returns the workspace of this Nuts Exception.

```
[read-only] public NutsWorkspace workspace
private final NutsWorkspace workspace
public NutsWorkspace getWorkspace()
```

 [? NutsExecutionException](#)

`public net.vpc.app.nuts.NutsExecutionException`

Standard Execution thrown when an artifact fails to run.

 Constant Fields

 DEFAULT_ERROR_EXIT_CODE

```
public static final int DEFAULT_ERROR_EXIT_CODE = 244
```

 Constructors

 NutsExecutionException(workspace)

Constructs a new NutsExecutionException exception

```
NutsExecutionException(NutsWorkspace workspace)
```

workspace

 NutsExecutionException(workspace, exitCode)

Constructs a new NutsExecutionException exception

```
NutsExecutionException(NutsWorkspace workspace, int exitCode)
```

workspace exit code

 NutsExecutionException(workspace, cause, exitCode)

Constructs a new NutsExecutionException exception

```
NutsExecutionException(NutsWorkspace workspace, Throwable cause, int exitCode)
```

workspace cause exit code

 NutsExecutionException(workspace, message, cause)

Constructs a new NutsExecutionException exception

```
NutsExecutionException(NutsWorkspace workspace, String message, Throwable cause)
```

workspace message cause

[¶ NutsExecutionException\(workspace, message, exitCode\)](#)

Constructs a new NutsExecutionException exception

```
NutsExecutionException(NutsWorkspace workspace, String message, int exitCode)
```

workspace message exit code

[¶ NutsExecutionException\(workspace, message, cause, exitCode\)](#)

Constructs a new NutsExecutionException exception

```
NutsExecutionException(NutsWorkspace workspace, String message, Throwable cause,  
int exitCode)
```

workspace message cause exit code

[¶ NutsExecutionException\(workspace, message, cause, enableSuppression, writableStackTrace,
exitCode\)](#)

Constructs a new NutsExecutionException exception

```
NutsExecutionException(NutsWorkspace workspace, String message, Throwable cause,  
boolean enableSuppression, boolean writableStackTrace, int exitCode)
```

workspace message cause whether or not suppression is enabled or disabled whether or not the
stack trace should be writable exit code

[¶ Instance Properties](#)

 [exitCode](#)

artifact exit code

```
[read-only] public int exitCode  
private final int exitCode  
public int getExitCode()
```

⌚ NutsExtensionAlreadyRegisteredException

```
public net.vpc.app.nuts.NutsExtensionAlreadyRegisteredException
```

Exception thrown when extension is already registered.

Constructors

⌚ NutsExtensionAlreadyRegisteredException(workspace, id, installed)

Constructs a new NutsExtensionAlreadyRegisteredException exception

```
NutsExtensionAlreadyRegisteredException(NutsWorkspace workspace, String id,  
String installed)
```

workspace artifact id installed id

⌚ NutsExtensionAlreadyRegisteredException(workspace, id, installed, cause)

Constructs a new NutsExtensionAlreadyRegisteredException exception

```
NutsExtensionAlreadyRegisteredException(NutsWorkspace workspace, String id,  
String installed, Throwable cause)
```

workspace artifact id installed id cause

Instance Properties

⌚ installed

installed id

```
[read-only] public String installed
private final String installed
public String getInstalled()
```

⌚ NutsExtensionException

```
public abstract net.vpc.app.nuts.NutsExtensionException
```

Base exception for Extension related exceptions

Constructors

⌚ NutsExtensionException(workspace, extensionId, message, cause)

Constructs a new runtime exception with the specified detail message and cause.

Note that the detail message associated with `cause` is not automatically incorporated in this runtime exception's detail message.

```
NutsExtensionException(NutsWorkspace workspace, String extensionId, String
message, Throwable cause)
```

the workspace of this Nuts Exception extension id the detail message (which is saved for later retrieval by the {@link #getMessage()} method). if the message is null, a default one is provided the cause (which is saved for later retrieval by the {@link #getCause()} method). (A <tt>null</tt> value is permitted, and indicates that the cause is nonexistent or unknown.)

Instance Properties

📄⌚ id

id

```
[read-only] public String id
private final String id
public String getId()
```

NutsExtensionNotFoundException

```
public net.vpc.app.nuts.NutsExtensionNotFoundException
```

Exception thrown when extension could not be resolved.

Constructors

NutsExtensionNotFoundException(workspace, missingType, extensionName)

Constructs a new NutsExtensionNotFoundException exception

```
NutsExtensionNotFoundException(NutsWorkspace workspace, Class missingType, String extensionName)
```

workspace missing type extension name

Instance Properties

extensionName

extension name

```
[read-only] public String extensionName  
private final String extensionName  
public String getExtensionName()
```

missingType

missing type

```
[read-only] public Class missingType  
private final Class missingType  
public Class getMissingType()
```

NutsFactoryException

```
public net.vpc.app.nuts.NutsFactoryException
```

Exception thrown when a component cannot be resolved by the factory.

[Constructors](#)

[NutsFactoryException\(workspace\)](#)

Constructs a new NutsFactoryException exception

```
NutsFactoryException(NutsWorkspace workspace)
```

workspace

[NutsFactoryException\(workspace, cause\)](#)

Constructs a new NutsFactoryException exception

```
NutsFactoryException(NutsWorkspace workspace, Throwable cause)
```

workspace cause

[NutsFactoryException\(workspace, cause\)](#)

Constructs a new NutsFactoryException exception

```
NutsFactoryException(NutsWorkspace workspace, IOException cause)
```

workspace cause

[NutsFactoryException\(workspace, message\)](#)

Constructs a new NutsFactoryException exception

```
NutsFactoryException(NutsWorkspace workspace, String message)
```

workspace message

[NutsFactoryException\(workspace, message, cause\)](#)

Constructs a new NutsFactoryException exception

```
NutsFactoryException(NutsWorkspace workspace, String message, Throwable cause)
```

workspace message cause

[NutsFactoryException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsFactoryException exception

```
NutsFactoryException(NutsWorkspace workspace, String message, Throwable cause,  
boolean enableSuppression, boolean writableStackTrace)
```

workspace message cause cause whether or not suppression is enabled or disabled whether or not
the stack trace should be writable

 [NutsFetchModeNotSupportedException](#)

```
public net.vpc.app.nuts.NutsFetchModeNotSupportedException
```

Created by vpc on 1/15/17.

[Constructors](#)

[NutsFetchModeNotSupportedException\(workspace, repo, fetchMode, id, message\)](#)

Constructs a new NutsFetchModeNotSupportedException exception

```
NutsFetchModeNotSupportedException(NutsWorkspace workspace, NutsRepository repo,  
NutsFetchMode fetchMode, String id, String message)
```

workspace repository fetch mode artifact id message

[? NutsFetchModeNotSupportedException\(workspace, repo, fetchMode, id, message, cause\)](#)

Constructs a new NutsFetchModeNotSupportedException exception

```
NutsFetchModeNotSupportedException(NutsWorkspace workspace, NutsRepository repo,  
NutsFetchMode fetchMode, String id, String message, Exception cause)
```

workspace repository fetch mode artifact id message cause

[? Instance Properties](#)

 [? fetchMode](#)

fetch mode

```
[read-only] public NutsFetchMode fetchMode  
private final NutsFetchMode fetchMode  
public NutsFetchMode getFetchMode()
```

 [? id](#)

artifact id

```
[read-only] public String id  
private final String id  
public String getId()
```

 [? repositoryName](#)

repository name

```
[read-only] public String repositoryName  
private final String repositoryName  
public String getRepositoryName()
```

[repositoryUuid](#)

repository uuid

```
[read-only] public String repositoryUuid  
private final String repositoryUuid  
public String getRepositoryUuid()
```

[NutsIllegalArgumentException](#)

```
public net.vpc.app.nuts.NutsIllegalArgumentException
```

Thrown to indicate that a method has been passed an illegal or inappropriate argument.

[Constructors](#)

[NutsIllegalArgumentException\(workspace\)](#)

Constructs a new NutsIllegalArgumentException exception

```
NutsIllegalArgumentException(NutsWorkspace workspace)
```

workspace

[NutsIllegalArgumentException\(workspace, cause\)](#)

Constructs a new NutsIllegalArgumentException exception

```
NutsIllegalArgumentException(NutsWorkspace workspace, Throwable cause)
```

workspace cause

[NutsIllegalArgumentException\(workspace, message\)](#)

Constructs a new NutsIllegalArgumentException exception

```
NutsIllegalArgumentException(NutsWorkspace workspace, String message)
```

workspace message

[NutsIllegalArgumentException\(workspace, message, cause\)](#)

Constructs a new NutsIllegalArgumentException exception

```
NutsIllegalArgumentException(NutsWorkspace workspace, String message, Throwable cause)
```

workspace message cause

[NutsIllegalArgumentException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsIllegalArgumentException exception

```
NutsIllegalArgumentException(NutsWorkspace workspace, String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace)
```

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

[NutsInstallException](#)

```
public net.vpc.app.nuts.NutsInstallException
```

Created by vpc on 1/15/17.

[Constructors](#)

[NutsInstallException\(workspace, id\)](#)

Custom Constructor

```
NutsInstallException(NutsWorkspace workspace, NutsId id)
```

workspace nuts id

[NutsInstallException\(workspace, id\)](#)

Custom Constructor

```
NutsInstallException(NutsWorkspace workspace, String id)
```

workspace nuts id

[NutsInstallException\(workspace, id, msg, ex\)](#)

Custom Constructor

```
NutsInstallException(NutsWorkspace workspace, NutsId id, String msg, Exception ex)
```

workspace nuts id message exception

[NutsInstallException\(workspace, id, msg, ex\)](#)

Custom Constructor

```
NutsInstallException(NutsWorkspace workspace, String id, String msg, Exception ex)
```

workspace nuts id message exception

 [NutsInstallationException](#)

```
public abstract net.vpc.app.nuts.NutsInstallationException
```

Base exception for installation fails.

Constructors

[NutsInstallationException\(workspace, id, msg, ex\)](#)

Custom Constructor

```
NutsInstallationException(NutsWorkspace workspace, String id, String msg,  
Exception ex)
```

workspace nuts id message exception

[NutsInstallationException\(workspace, id, msg, ex\)](#)

Custom Constructor

```
NutsInstallationException(NutsWorkspace workspace, NutsId id, String msg,  
Exception ex)
```

workspace nuts id message exception

Instance Properties

[id](#)

artifact id

```
[read-only] public String id  
private final String id  
public String getId()
```

[NutsInvalidRepositoryException](#)

```
public net.vpc.app.nuts.NutsInvalidRepositoryException
```

This Exception is thrown when the repository fails to initialize.

Constructors

[NutsInvalidRepositoryException\(workspace, repository, message\)](#)

Constructs a new NutsInvalidRepositoryException exception

```
NutsInvalidRepositoryException(NutsWorkspace workspace, String repository, String  
message)
```

workspace repository message

NutsInvalidWorkspaceException

```
public net.vpc.app.nuts.NutsInvalidWorkspaceException
```

This Exception is thrown when the workspace fails to initialize.

Constructors

[NutsInvalidWorkspaceException\(workspace, workspaceLocation, errorMessage\)](#)

Constructs a new NutsInvalidWorkspaceException exception

```
NutsInvalidWorkspaceException(NutsWorkspace workspace, String workspaceLocation,  
String errorMessage)
```

workspace workspaceLocation errorMessage

Instance Properties

workspaceLocation

workspace location

```
[read-only] public String workspaceLocation  
private final String workspaceLocation  
public String getWorkspaceLocation()
```

NutsLockAcquireException

```
public net.vpc.app.nuts.NutsLockAcquireException
```

Exception Thrown when a locked object is invoked.

Constructors

NutsLockAcquireException(workspace, lockedObject, lockObject)

Constructs a new ock exception.

```
NutsLockAcquireException(NutsWorkspace workspace, Object lockedObject, Object  
lockObject)
```

workspace locked object lock Object

NutsLockAcquireException(workspace, message, lockedObject, lockObject)

Constructs a new ock exception.

```
NutsLockAcquireException(NutsWorkspace workspace, String message, Object  
lockedObject, Object lockObject)
```

workspace message or null locked Object lock Object

NutsLockAcquireException(workspace, message, lockedObject, lockObject, cause)

Constructs a new ock exception.

```
NutsLockAcquireException(NutsWorkspace workspace, String message, Object  
lockedObject, Object lockObject, Throwable cause)
```

workspace message or null locked Object lock Object cause

☕ [NutsLockException](#)

```
public net.vpc.app.nuts.NutsLockException
```

Exception Thrown when a locked object is invoked.

Constructors

⌚ [NutsLockException\(workspace, lockedObject, lockObject\)](#)

Constructs a new ock exception.

```
NutsLockException(NutsWorkspace workspace, Object lockedObject, Object  
lockObject)
```

workspace locked Object lock Object

⌚ [NutsLockException\(workspace, message, lockedObject, lockObject\)](#)

Constructs a new ock exception.

```
NutsLockException(NutsWorkspace workspace, String message, Object lockedObject,  
Object lockObject)
```

workspace message or null locked Object lock Object

⌚ [NutsLockException\(workspace, message, lockedObject, lockObject, cause\)](#)

Constructs a new ock exception.

```
NutsLockException(NutsWorkspace workspace, String message, Object lockedObject,
Object lockObject, Throwable cause)
```

workspace message or null locked Object lock Object cause

[Instance Properties](#)

 [lockObject](#)

lock object

```
[read-only] public Object lockObject
private Object lockObject
public Object getLockObject\(\)
```

 [lockedObject](#)

locked object

```
[read-only] public Object lockedObject
private Object lockedObject
public Object getLockedObject\(\)
```

[NutsLoginException](#)

```
public net.vpc.app.nuts.NutsLoginException
```

[Constructors](#)

[NutsLoginException\(workspace\)](#)

Constructs a new NutsLoginException exception

```
NutsLoginException(NutsWorkspace workspace)
```

workspace

[🔗 NutsLoginException\(workspace, cause\)](#)

Constructs a new NutsLoginException exception

`NutsLoginException(NutsWorkspace workspace, Throwable cause)`

workspace cause

[🔗 NutsLoginException\(workspace, message\)](#)

Constructs a new NutsLoginException exception

`NutsLoginException(NutsWorkspace workspace, String message)`

workspace message

[🔗 NutsLoginException\(workspace, message, cause\)](#)

Constructs a new NutsLoginException exception

`NutsLoginException(NutsWorkspace workspace, String message, Throwable cause)`

workspace message cause

[🔗 NutsLoginException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsLoginException exception

`NutsLoginException(NutsWorkspace workspace, String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace)`

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

[⚠️ NutsNotExecutableException](#)

```
public net.vpc.app.nuts.NutsNotExecutableException
```

Exception thrown when a non executable nuts id is requested to run.

Constructors

[NutsNotExecutableException\(workspace, id\)](#)

Constructs a new NutsNotExecutableException exception

```
NutsNotExecutableException(NutsWorkspace workspace, NutsId id)
```

workspace artifact id

[NutsNotExecutableException\(workspace, id\)](#)

Constructs a new NutsNotExecutableException exception

```
NutsNotExecutableException(NutsWorkspace workspace, String id)
```

workspace artifact id

Instance Properties

 [id](#)

artifact id

```
[read-only] public String id  
private final String id  
public String getId()
```

 [NutsNotFoundException](#)

```
public net.vpc.app.nuts.NutsNotFoundException
```

Exception thrown when the package could not be resolved

[Constructors](#)

[NutsNotFoundException\(workspace, id\)](#)

Constructs a new NutsNotFoundException exception

```
NutsNotFoundException(NutsWorkspace workspace, NutsId id)
```

workspace artifact id

[NutsNotFoundException\(workspace, id\)](#)

Constructs a new NutsNotFoundException exception

```
NutsNotFoundException(NutsWorkspace workspace, String id)
```

workspace artifact id

[NutsNotFoundException\(workspace, id, cause\)](#)

Constructs a new NutsNotFoundException exception

```
NutsNotFoundException(NutsWorkspace workspace, NutsId id, Exception cause)
```

workspace artifact id cause

[NutsNotFoundException\(workspace, id, message, cause\)](#)

Constructs a new NutsNotFoundException exception

```
NutsNotFoundException(NutsWorkspace workspace, String id, String message,
Exception cause)
```

workspace artifact id message cause

[NutsNotFoundException\(workspace, id, message, cause\)](#)

Constructs a new NutsNotFoundException exception

```
NutsNotFoundException(NutsWorkspace workspace, NutsId id, String message,  
Exception cause)
```

workspace artifact id message cause

[Instance Properties](#) [id](#)

artifact id

```
[read-only] public String id  
private final String id  
public String getId()
```

 [NutsNotInstallableException](#)

```
public net.vpc.app.nuts.NutsNotInstallableException
```

This exception is thrown when an artifact fails to be installed.

[Constructors](#)[NutsNotInstallableException\(workspace, id\)](#)

Constructs a new NutsNotInstallableException exception

```
NutsNotInstallableException(NutsWorkspace workspace, NutsId id)
```

workspace artifact

[NutsNotInstallableException\(workspace, id\)](#)

Constructs a new NutsNotInstallableException exception

```
NutsNotInstallableException(NutsWorkspace workspace, String id)
```

workspace artifact

[NutsNotInstallableException\(workspace, id, msg, ex\)](#)

Constructs a new NutsNotInstallableException exception

```
NutsNotInstallableException(NutsWorkspace workspace, NutsId id, String msg,  
Exception ex)
```

workspace artifact message exception

[NutsNotInstallableException\(workspace, id, msg, ex\)](#)

Constructs a new NutsNotInstallableException exception

```
NutsNotInstallableException(NutsWorkspace workspace, String id, String msg,  
Exception ex)
```

workspace artifact message exception

[NutsNotInstalledException](#)

```
public net.vpc.app.nuts.NutsNotInstalledException
```

This Exception is fired when an artifact fails to be uninstalled for the artifact not being installed yet.

[Constructors](#)

[NutsNotInstalledException\(workspace, id\)](#)

Constructs a new NutsNotInstalledException exception

```
NutsNotInstalledException(NutsWorkspace workspace, NutsId id)
```

workspace artifact

[🔗 NutsNotInstalledException\(workspace, id\)](#)

Constructs a new NutsNotInstalledException exception

```
NutsNotInstalledException(NutsWorkspace workspace, String id)
```

workspace artifact

[🔗 NutsNotInstalledException\(workspace, id, msg, ex\)](#)

Constructs a new NutsNotInstalledException exception

```
NutsNotInstalledException(NutsWorkspace workspace, NutsId id, String msg,  
Exception ex)
```

workspace artifact message error

[🔗 NutsNotInstalledException\(workspace, id, msg, ex\)](#)

Constructs a new NutsNotInstalledException exception

```
NutsNotInstalledException(NutsWorkspace workspace, String id, String msg,  
Exception ex)
```

workspace artifact message exception

 [NutsParseEnumException](#)

```
public net.vpc.app.nuts.NutsParseEnumException
```

Exception Thrown when for any reason, the enum value is not expected/supported.

Constructors

[NutsParseEnumException\(workspace, invalidValue, enumType\)](#)

create new instance of NutsUnexpectedEnumException

```
NutsParseEnumException(NutsWorkspace workspace, String invalidValue, Class<? extends Enum> enumType)
```

workspace invalid value enumeration instance (cannot be null)

[NutsParseEnumException\(workspace, message, invalidValue, enumType\)](#)

create new instance of NutsUnexpectedEnumException

```
NutsParseEnumException(NutsWorkspace workspace, String message, String invalidValue, Class<? extends Enum> enumType)
```

workspace message invalid value enumeration instance (cannot be null)

Instance Properties

 [enumType](#)

enum type

```
[read-only] public Class<? extends Enum> enumType  
private Class<? extends Enum> enumType  
public Class<? extends Enum> getEnumType()
```

 [invalidValue](#)

return invalid value

```
[read-only] public String invalidValue  
private String invalidValue  
public String getInvalidValue()
```

 NutsParseException

```
public net.vpc.app.nuts.NutsParseException
```

 [Constructors](#) [NutsParseException\(workspace\)](#)

Constructs a new NutsParseException exception

```
NutsParseException(NutsWorkspace workspace)
```

workspace

 [NutsParseException\(workspace, cause\)](#)

Constructs a new NutsParseException exception

```
NutsParseException(NutsWorkspace workspace, Throwable cause)
```

workspace cause

 [NutsParseException\(workspace, message\)](#)

Constructs a new NutsParseException exception

```
NutsParseException(NutsWorkspace workspace, String message)
```

workspace message

 [NutsParseException\(workspace, message, cause\)](#)

Constructs a new NutsParseException exception

```
NutsParseException(NutsWorkspace workspace, String message, Throwable cause)
```

workspace message cause

[NutsParseException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsParseException exception

```
NutsParseException(NutsWorkspace workspace, String message, Throwable cause,  
boolean enableSuppression, boolean writableStackTrace)
```

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

 [NutsPushException](#)

```
public net.vpc.app.nuts.NutsPushException
```

Push Exception

[Constructors](#)

[NutsPushException\(workspace, id\)](#)

Constructs a new NutsPushException exception

```
NutsPushException(NutsWorkspace workspace, String id)
```

workspace artifact id

[NutsPushException\(workspace, id\)](#)

Constructs a new NutsPushException exception

```
NutsPushException(NutsWorkspace workspace, NutsId id)
```

workspace artifact id

[NutsPushException\(workspace, id, message\)](#)

Constructs a new NutsPushException exception

```
NutsPushException(NutsWorkspace workspace, String id, String message)
```

workspace artifact id message

[NutsPushException\(workspace, id, message\)](#)

Constructs a new NutsPushException exception

```
NutsPushException(NutsWorkspace workspace, NutsId id, String message)
```

workspace artifact id message

[NutsPushException\(workspace, id, message, cause\)](#)

Constructs a new NutsPushException exception

```
NutsPushException(NutsWorkspace workspace, String id, String message, Throwable cause)
```

workspace artifact id message cause

[NutsPushException\(workspace, id, message, cause\)](#)

Constructs a new NutsPushException exception

```
NutsPushException(NutsWorkspace workspace, NutsId id, String message, Throwable cause)
```

workspace artifact id message cause

[Instance Properties](#)

[id](#)

artifact id

```
[read-only] public String id  
private final String id  
public String getId()
```

[NutsReadOnlyException](#)

```
public net.vpc.app.nuts.NutsReadOnlyException
```

Created by vpc on 1/15/17.

[Constructors](#)

[NutsReadOnlyException\(workspace\)](#)

Constructs a new NutsReadOnlyException exception

```
NutsReadOnlyException(NutsWorkspace workspace)
```

workspace

[NutsReadOnlyException\(workspace, location\)](#)

Constructs a new NutsReadOnlyException exception

```
NutsReadOnlyException(NutsWorkspace workspace, String location)
```

workspace location

[NutsRepositoryAlreadyRegisteredException](#)

```
public net.vpc.app.nuts.NutsRepositoryAlreadyRegisteredException
```

This exception is thrown when a repository location could no be loaded because the repository is already registered for the actual workspace.

Constructors

[NutsRepositoryAlreadyRegisteredException\(workspace, repository\)](#)

Constructs a new NutsNotInstalledException exception

```
NutsRepositoryAlreadyRegisteredException(NutsWorkspace workspace, String  
repository)
```

workspace repository

[NutsRepositoryAlreadyRegisteredException\(workspace, repository, err\)](#)

Constructs a new NutsNotInstalledException exception

```
NutsRepositoryAlreadyRegisteredException(NutsWorkspace workspace, String  
repository, Throwable err)
```

workspace repository error

NutsRepositoryException

```
public abstract net.vpc.app.nuts.NutsRepositoryException
```

Base exception for Repository related exceptions

Constructors

[NutsRepositoryException\(workspace, repository, message, ex\)](#)

Constructs a new NutsRepositoryException exception

```
NutsRepositoryException(NutsWorkspace workspace, String repository, String  
message, Throwable ex)
```

workspace repository message exception

[Instance Properties](#)

[repository](#)

the repository of this exception

```
[read-only] public String repository  
private final String repository  
public String getRepository()
```

[NutsRepositoryNotFoundException](#)

```
public net.vpc.app.nuts.NutsRepositoryNotFoundException
```

This exception is thrown when a repository location could no be loaded because the repository config files are missing.

[Constructors](#)

[NutsRepositoryNotFoundException\(workspace, repository\)](#)

Constructs a new NutsRepositoryNotFoundException exception

```
NutsRepositoryNotFoundException(NutsWorkspace workspace, String repository)
```

workspace repository

[NutsSecurityException](#)

```
public net.vpc.app.nuts.NutsSecurityException
```

Thrown by Nuts Workspace to indicate a security violation.

 [Constructors](#) [NutsSecurityException\(workspace\)](#)

Constructs a `` NutsSecurityException

parameters.

 [NutsSecurityException\(NutsWorkspace workspace\)](#)

workspace

```
####  NutsSecurityException\(workspace, cause\)
Constructs a ``
NutsSecurityException
```

parameters.

 [NutsSecurityException\(NutsWorkspace workspace, Throwable cause\)](#)

workspace cause

 [NutsSecurityException\(workspace, message\)](#)

Constructs a `` NutsSecurityException

parameters.

 [NutsSecurityException\(NutsWorkspace workspace, String message\)](#)

workspace
the detail message.

```
####  NutsSecurityException\(workspace, message, cause\)
Constructs a ``
NutsSecurityException
```

parameters.

```
NutsSecurityException(NutsWorkspace workspace, String message, Throwable cause)
```

workspace message cause

[Instance Properties](#)

  workspace

workspace

```
[read-only] public NutsWorkspace workspace  
private NutsWorkspace workspace  
public NutsWorkspace getWorkspace()
```

[NutsTooManyElementsException](#)

```
public net.vpc.app.nuts.NutsTooManyElementsException
```

[Constructors](#)

[NutsTooManyElementsException\(workspace\)](#)

Constructs a new NutsTooManyElementsException exception

```
NutsTooManyElementsException(NutsWorkspace workspace)
```

workspace

[NutsTooManyElementsException\(workspace, cause\)](#)

Constructs a new NutsTooManyElementsException exception

```
NutsTooManyElementsException(NutsWorkspace workspace, Throwable cause)
```

workspace cause

[NutsTooManyElementsException\(workspace, cause\)](#)

Constructs a new NutsTooManyElementsException exception

NutsTooManyElementsException(NutsWorkspace workspace, IOException cause)

workspace cause

[NutsTooManyElementsException\(workspace, message\)](#)

Constructs a new NutsTooManyElementsException exception

NutsTooManyElementsException(NutsWorkspace workspace, String message)

workspace message

[NutsTooManyElementsException\(workspace, message, cause\)](#)

Constructs a new NutsTooManyElementsException exception

NutsTooManyElementsException(NutsWorkspace workspace, String message, Throwable cause)

workspace message cause

[NutsTooManyElementsException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsTooManyElementsException exception

NutsTooManyElementsException(NutsWorkspace workspace, String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace)

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

⌚ NutsUnexpectedException

```
public net.vpc.app.nuts.NutsUnexpectedException
```

Constructors

NutsUnexpectedException(workspace)

Constructs a new NutsUnexpectedException exception

```
NutsUnexpectedException(NutsWorkspace workspace)
```

workspace

NutsUnexpectedException(workspace, cause)

Constructs a new NutsUnexpectedException exception

```
NutsUnexpectedException(NutsWorkspace workspace, Throwable cause)
```

workspace cause

NutsUnexpectedException(workspace, cause)

Constructs a new NutsUnexpectedException exception

```
NutsUnexpectedException(NutsWorkspace workspace, IOException cause)
```

workspace cause

NutsUnexpectedException(workspace, message)

Constructs a new NutsUnexpectedException exception

```
NutsUnexpectedException(NutsWorkspace workspace, String message)
```

workspace message

[🔗 NutsUnexpectedException\(workspace, message, cause\)](#)

Constructs a new NutsUnexpectedException exception

```
NutsUnexpectedException(NutsWorkspace workspace, String message, Throwable cause)
```

workspace message cause

[🔗 NutsUnexpectedException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsUnexpectedException exception

```
NutsUnexpectedException(NutsWorkspace workspace, String message, Throwable cause,  
boolean enableSuppression, boolean writableStackTrace)
```

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

[🔗 NutsUninstallException](#)

```
public net.vpc.app.nuts.NutsUninstallException
```

This Exception is thrown when an artifact fails to be uninstalled

[🔗 Constructors](#)

[🔗 NutsUninstallException\(workspace, id\)](#)

Custom Constructor

```
NutsUninstallException(NutsWorkspace workspace, NutsId id)
```

workspace nuts id

[NutsUninstallException\(workspace, id\)](#)

Custom Constructor

```
NutsUninstallException(NutsWorkspace workspace, String id)
```

workspace nuts id

[NutsUninstallException\(workspace, id, msg, ex\)](#)

Custom Constructor

```
NutsUninstallException(NutsWorkspace workspace, NutsId id, String msg, Exception ex)
```

workspace nuts id message exception

[NutsUninstallException\(workspace, id, msg, ex\)](#)

Custom Constructor

```
NutsUninstallException(NutsWorkspace workspace, String id, String msg, Exception ex)
```

workspace nuts id message exception

[NutsUnsatisfiedRequirementsException](#)

```
public net.vpc.app.nuts.NutsUnsatisfiedRequirementsException
```

[Constructors](#)

[NutsUnsatisfiedRequirementsException\(workspace\)](#)

Constructs a new NutsUnsatisfiedRequirementsException exception

NutsUnsatisfiedRequirementsException(NutsWorkspace workspace)

workspace

[¶ NutsUnsatisfiedRequirementsException\(workspace, cause\)](#)

Constructs a new NutsUnsatisfiedRequirementsException exception

NutsUnsatisfiedRequirementsException(NutsWorkspace workspace, Throwable cause)

workspace cause

[¶ NutsUnsatisfiedRequirementsException\(workspace, cause\)](#)

Constructs a new NutsUnsatisfiedRequirementsException exception

NutsUnsatisfiedRequirementsException(NutsWorkspace workspace, IOException cause)

workspace cause

[¶ NutsUnsatisfiedRequirementsException\(workspace, message\)](#)

Constructs a new NutsUnsatisfiedRequirementsException exception

NutsUnsatisfiedRequirementsException(NutsWorkspace workspace, String message)

workspace message

[¶ NutsUnsatisfiedRequirementsException\(workspace, message, cause\)](#)

Constructs a new NutsUnsatisfiedRequirementsException exception

NutsUnsatisfiedRequirementsException(NutsWorkspace workspace, String message, Throwable cause)

workspace message cause

[NutsUnsatisfiedRequirementsException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsUnsatisfiedRequirementsException exception

```
NutsUnsatisfiedRequirementsException(NutsWorkspace workspace, String message,  
Throwable cause, boolean enableSuppression, boolean writableStackTrace)
```

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

 [NutsUnsupportedArgumentException](#)

```
public net.vpc.app.nuts.NutsUnsupportedArgumentException
```

[Constructors](#)

[NutsUnsupportedArgumentException\(workspace\)](#)

Constructs a new NutsUnsupportedArgumentException exception

```
NutsUnsupportedArgumentException(NutsWorkspace workspace)
```

workspace

[NutsUnsupportedArgumentException\(workspace, cause\)](#)

Constructs a new NutsUnsupportedArgumentException exception

```
NutsUnsupportedArgumentException(NutsWorkspace workspace, Throwable cause)
```

workspace cause

[NutsUnsupportedArgumentException\(workspace, message\)](#)

Constructs a new NutsUnsupportedArgumentException exception

```
NutsUnsupportedArgumentException(NutsWorkspace workspace, String message)
```

workspace message

[? NutsUnsupportedArgumentException\(workspace, message, cause\)](#)

Constructs a new NutsUnsupportedArgumentException exception

```
NutsUnsupportedArgumentException(NutsWorkspace workspace, String message,  
Throwable cause)
```

workspace message cause

[? NutsUnsupportedArgumentException\(workspace, message, cause, enableSuppression,
writableStackTrace\)](#)

Constructs a new NutsUnsupportedArgumentException exception

```
NutsUnsupportedArgumentException(NutsWorkspace workspace, String message,  
Throwable cause, boolean enableSuppression, boolean writableStackTrace)
```

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

 [NutsUnsupportedEnumException](#)

```
public net.vpc.app.nuts.NutsUnsupportedEnumException
```

Exception Thrown when for any reason, the enum value is not expected/supported.

[? Constructors](#)

[? NutsUnsupportedEnumException\(workspace, enumValue\)](#)

create new instance of NutsUnexpectedEnumException

```
NutsUnsupportedEnumException(NutsWorkspace workspace, Enum enumValue)
```

workspace enumeration instance (cannot be null)

[🔗 NutsUnsupportedEnumException\(workspace, message, enumValue\)](#)

create new instance of NutsUnexpectedEnumException

```
NutsUnsupportedEnumException(NutsWorkspace workspace, String message, Enum enumValue)
```

workspace message enumeration instance (cannot be null)

[🔗 NutsUnsupportedEnumException\(workspace, message, stringValue, enumValue\)](#)

create new instance of NutsUnexpectedEnumException

```
NutsUnsupportedEnumException(NutsWorkspace workspace, String message, String stringValue, Enum enumValue)
```

workspace message invalid value enumeration instance (cannot be null)

[🔗 Instance Properties](#)

 [🔗 enumValue](#)

enum value

```
[read-only] public Enum enumValue  
private Enum enumValue  
public Enum getEnumValue()
```

 [🔗 NutsUnsupportedOperationException](#)

```
public net.vpc.app.nuts.NutsUnsupportedOperationException
```

 [Constructors](#) [NutsUnsupportedOperationException\(workspace\)](#)

Constructs a new NutsUnsupportedOperationException exception

```
NutsUnsupportedOperationException(NutsWorkspace workspace)
```

workspace

 [NutsUnsupportedOperationException\(workspace, cause\)](#)

Constructs a new NutsUnsupportedOperationException exception

```
NutsUnsupportedOperationException(NutsWorkspace workspace, Throwable cause)
```

workspace cause

 [NutsUnsupportedOperationException\(workspace, message\)](#)

Constructs a new NutsUnsupportedOperationException exception

```
NutsUnsupportedOperationException(NutsWorkspace workspace, String message)
```

workspace message

 [NutsUnsupportedOperationException\(workspace, message, cause\)](#)

Constructs a new NutsUnsupportedOperationException exception

```
NutsUnsupportedOperationException(NutsWorkspace workspace, String message, Throwable cause)
```

workspace message cause

[` NutsUnsupportedOperationException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsUnsupportedOperationException exception

```
NutsUnsupportedOperationException(NutsWorkspace workspace, String message,  
Throwable cause, boolean enableSuppression, boolean writableStackTrace)
```

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

[` NutsUserCancelException](#)

```
public net.vpc.app.nuts.NutsUserCancelException
```

[` Constant Fields](#)

[` DEFAULT_CANCEL_EXIT_CODE](#)

```
public static final int DEFAULT_CANCEL_EXIT_CODE = 245
```

[` Constructors](#)

[` NutsUserCancelException\(workspace\)](#)

Constructs a new NutsUserCancelException exception

```
NutsUserCancelException(NutsWorkspace workspace)
```

workspace

[` NutsUserCancelException\(workspace, message\)](#)

Constructs a new NutsUserCancelException exception

```
NutsUserCancelException(NutsWorkspace workspace, String message)
```

workspace message

[¶ NutsUserCancelException\(workspace, message, exitCode\)](#)

Constructs a new NutsUserCancelException exception

```
NutsUserCancelException(NutsWorkspace workspace, String message, int exitCode)
```

workspace message exit code

[↳ NutsValidationException](#)

```
public net.vpc.app.nuts.NutsValidationException
```

[¶ Constructors](#)

[¶ NutsValidationException\(workspace\)](#)

Constructs a new NutsValidationException exception

```
NutsValidationException(NutsWorkspace workspace)
```

workspace

[¶ NutsValidationException\(workspace, cause\)](#)

Constructs a new NutsValidationException exception

```
NutsValidationException(NutsWorkspace workspace, Throwable cause)
```

workspace cause

[NutsValidationException\(workspace, cause\)](#)

Constructs a new NutsValidationException exception

```
NutsValidationException(NutsWorkspace workspace, IOException cause)
```

workspace cause

[NutsValidationException\(workspace, message\)](#)

Constructs a new NutsValidationException exception

```
NutsValidationException(NutsWorkspace workspace, String message)
```

workspace message

[NutsValidationException\(workspace, message, cause\)](#)

Constructs a new NutsValidationException exception

```
NutsValidationException(NutsWorkspace workspace, String message, Throwable cause)
```

workspace message cause

[NutsValidationException\(workspace, message, cause, enableSuppression, writableStackTrace\)](#)

Constructs a new NutsValidationException exception

```
NutsValidationException(NutsWorkspace workspace, String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace)
```

workspace message cause whether or not suppression is enabled or disabled whether or not the stack trace should be writable

[NutsWorkspaceAlreadyExistsException](#)

```
public net.vpc.app.nuts.NutsWorkspaceAlreadyExistsException
```

Created by vpc on 1/15/17.

Constructors

[NutsWorkspaceAlreadyExistsException\(workspace, workspaceLocation\)](#)

Constructs a new NutsWorkspaceAlreadyExistsException exception

```
NutsWorkspaceAlreadyExistsException(NutsWorkspace workspace, String  
workspaceLocation)
```

workspace location

[NutsWorkspaceAlreadyExistsException\(workspace, workspaceLocation, err\)](#)

Constructs a new NutsWorkspaceAlreadyExistsException exception

```
NutsWorkspaceAlreadyExistsException(NutsWorkspace workspace, String  
workspaceLocation, Throwable err)
```

workspace location exception

Instance Properties

[workspaceLocation](#)

workspace location

```
[read-only] public String workspaceLocation  
private final String workspaceLocation  
public String getWorkspaceLocation()
```

[NutsWorkspaceException](#)

```
public abstract net.vpc.app.nuts.NutsWorkspaceException
```

NutsWorkspaceException is the base class for Workspace related exceptions.

Constructors

[NutsWorkspaceException\(workspace, message, ex\)](#)

Constructs a new NutsWorkspaceException exception

```
NutsWorkspaceException(NutsWorkspace workspace, String message, Throwable ex)
```

workspace message exception

NutsWorkspaceNotFoundException

```
public net.vpc.app.nuts.NutsWorkspaceNotFoundException
```

This Exception is thrown when the workspace does not exist.

Constructors

[NutsWorkspaceNotFoundException\(workspace, workspaceLocation\)](#)

Constructs a new NutsWorkspaceNotFoundException exception

```
NutsWorkspaceNotFoundException(NutsWorkspace workspace, String workspaceLocation)
```

workspace location

Instance Properties

[workspaceLocation](#)

workspace location

```
[read-only] public String workspaceLocation  
private final String workspaceLocation  
public String getWorkspaceLocation()
```

Extensions

☕ NutsRepositoryModel

```
public net.vpc.app.nuts.NutsRepositoryModel
```

⌚ Instance Fields

⌚ CACHE

```
int CACHE = CACHE_READ | CACHE_WRITE
```

⌚ CACHE_READ

```
int CACHE_READ = 16
```

⌚ CACHE_WRITE

```
int CACHE_WRITE = 32
```

⌚ LIB

```
int LIB = LIB_READ | LIB_WRITE | LIB_OVERRIDE
```

⌚ LIB_OVERRIDE

```
int LIB_OVERRIDE = 8
```

⌚ LIB_READ

```
int LIB_READ = 2
```

LIB_WRITE

```
int LIB_WRITE = 4
```

MIRRORING

```
int MIRRORING = 1
```

Instance Properties

deployOrder

```
[read-only] default int deployOrder  
default int getDeployOrder()
```

mode

```
[read-only] default int mode  
default int getMode()
```

name

```
[read-only] String name  
String getName()
```

repositoryType

```
[read-only] default String repositoryType  
default String getRepositoryType()
```

speed

```
[read-only] default int speed  
default int getSpeed()
```

📄 ↴ storeLocationStrategy

```
[read-only] default NutsStoreLocationStrategy storeLocationStrategy  
default NutsStoreLocationStrategy getStoreLocationStrategy()
```

📄 ↴ uuid

```
[read-only] default String uuid  
default String getUuid()
```

⚙️ Instance Methods

⚙️ acceptDeploy(id, mode, repository, session)

```
boolean acceptDeploy(NutsId id, NutsFetchMode mode, NutsRepository repository,  
NutsSession session)
```

null null null null

⚙️ acceptFetch(id, mode, repository, session)

```
boolean acceptFetch(NutsId id, NutsFetchMode mode, NutsRepository repository,  
NutsSession session)
```

null null null null

⚙️ fetchContent(id, descriptor, localPath, fetchMode, repository, session)

```
NutsContent fetchContent(NutsId id, NutsDescriptor descriptor, Path localPath,  
NutsFetchMode fetchMode, NutsRepository repository, NutsSession session)
```

null null null null null null

⚙️ `fetchDescriptor(id, fetchMode, repository, session)`

```
NutsDescriptor fetchDescriptor(NutsId id, NutsFetchMode fetchMode, NutsRepository repository, NutsSession session)
```

null null null null

⚙️ `search(filter, roots, fetchMode, repository, session)`

```
Iterator<NutsId> search(NutsIdFilter filter, String[] roots, NutsFetchMode fetchMode, NutsRepository repository, NutsSession session)
```

null null null null null

⚙️ `searchLatestVersion(id, filter, fetchMode, repository, session)`

```
NutsId searchLatestVersion(NutsId id, NutsIdFilter filter, NutsFetchMode fetchMode, NutsRepository repository, NutsSession session)
```

null null null null null

⚙️ `searchVersions(id, idFilter, fetchMode, repository, session)`

```
Iterator<NutsId> searchVersions(NutsId id, NutsIdFilter idFilter, NutsFetchMode fetchMode, NutsRepository repository, NutsSession session)
```

null null null null null

⚙️ `updateStatistics(repository, session)`

```
void updateStatistics(NutsRepository repository, NutsSession session)
```

null null

☕ NutsWorkspaceExtension

```
public net.vpc.app.nuts.NutsWorkspaceExtension
```

Created by vpc on 1/15/17.

⌚ Instance Properties

⌚ id

extension id pattern (configured)

```
[read-only] NutsId id  
NutsId getId()
```

⌚ wiredId

extension id resolved and wired

```
[read-only] NutsId wiredId  
NutsId getWiredId()
```

☕ NutsWorkspaceExtensionManager

```
public net.vpc.app.nuts.NutsWorkspaceExtensionManager
```

⌚ Instance Properties

⌚ extensionPoints

```
[read-only] Set<Class> extensionPoints  
Set<Class> getExtensionPoints()
```

extensions

return loaded extensions

```
[read-only] NutsId[] extensions  
NutsId[] getExtensions()
```

Instance Methods

createAll(type)

```
List<T> createAll(Class<T> type)
```

null

createAllSupported(type, supportCriteria)

```
List<T> createAllSupported(Class<T> type, NutsSupportLevelContext<V>  
supportCriteria)
```

null null

createServiceLoader(serviceType, criteriaType)

```
NutsServiceLoader<T,B> createServiceLoader(Class<T> serviceType, Class<B>  
criteriaType)
```

null null

createServiceLoader(serviceType, criteriaType, classLoader)

```
NutsServiceLoader<T,B> createServiceLoader(Class<T> serviceType, Class<B>  
criteriaType, ClassLoader classLoader)
```

null null null

⚙️ `createSupported(type, supportCriteria)`

create supported extension implementation or return null.

```
T createSupported(Class<T> type, NutsSupportLevelContext<V> supportCriteria)
```

extension type context

⚙️ `createSupported(type, supportCriteria, constructorParameterTypes, constructorParameters)`

create supported extension implementation or return null.

```
T createSupported(Class<T> type, NutsSupportLevelContext<V> supportCriteria,
Class[] constructorParameterTypes, Object[] constructorParameters)
```

extension type context constructor Parameter Types constructor Parameters

⚙️ `discoverTypes(classLoader)`

```
List<Class> discoverTypes(ClassLoader classLoader)
```

null

⚙️ `getExtensionObjects(extensionPoint)`

```
List<Object> getExtensionObjects(Class extensionPoint)
```

null

⚙️ `getExtensionTypes(extensionPoint)`

```
Set<Class> getExtensionTypes(Class extensionPoint)
```

null

⚙️ `getImplementationTypes(type)`

```
List<Class> getImplementationTypes(Class type)
```

null

⚙️ `installWorkspaceExtensionComponent(extensionPointType, extensionImpl)`

```
boolean installWorkspaceExtensionComponent(Class extensionPointType, Object extensionImpl)
```

null null

⚙️ `isRegisteredInstance(extensionPointType, extensionImpl)`

```
boolean isRegisteredInstance(Class extensionPointType, Object extensionImpl)
```

null null

⚙️ `isRegisteredType(extensionPointType, extensionType)`

```
boolean isRegisteredType(Class extensionPointType, Class extensionType)
```

null null

⚙️ `isRegisteredType(extensionPointType, name)`

```
boolean isRegisteredType(Class extensionPointType, String name)
```

null null

⚙️ `registerInstance(extensionPoint, implementation)`

```
boolean registerInstance(Class<T> extensionPoint, T implementation)
```

null null

⚙ registerType(extensionPointType, extensionType)

```
boolean registerType(Class extensionPointType, Class extensionType)
```

null null

Format

✍ NutsDependencyFormat

```
public net.vpc.app.nuts.NutsDependencyFormat
```

Dependency Format Helper

Instance Properties

✍ highlightImportedGroup

if true omit (do not include) name space when formatting the value set using `#setValue(NutsDependency)` .

```
[read-write] NutsDependencyFormat highlightImportedGroup  
boolean isHighlightImportedGroup()  
NutsDependencyFormat setHighlightImportedGroup(highlightImportedGroup)
```

✍ highlightOptional

if true omit (do not include) name space when formatting the value set using `#setValue(NutsDependency)` .

```
[read-write] NutsDependencyFormat highlightOptional  
boolean isHighlightOptional()  
NutsDependencyFormat setHighlightOptional(highlightOptional)
```

✍ highlightScope

if true omit (do not include) name space when formatting the value set using `#setValue(NutsDependency)` .

```
[read-write] NutsDependencyFormat highlightScope  
boolean isHighlightScope()  
NutsDependencyFormat setHighlightScope(highlightScope)
```

`omitClassifier`

if true omit (do not include) face when formatting the value set using `#setValue(NutsDependency)`.

```
[read-write] NutsDependencyFormat omitClassifier  
boolean isOmitClassifier()  
NutsDependencyFormat setOmitClassifier(value)
```

`omitExclusions`

if true omit (do not include) face when formatting the value set using `#setValue(NutsDependency)`.

```
[read-write] NutsDependencyFormat omitExclusions  
boolean isOmitExclusions()  
NutsDependencyFormat setOmitExclusions(value)
```

`omitGroupId`

if true omit (do not include) group when formatting the value set using `#setValue(NutsDependency)`.

```
[read-write] NutsDependencyFormat omitGroupId  
boolean isOmitGroupId()  
NutsDependencyFormat setOmitGroupId(omitGroup)
```

`omitImportedGroup`

if true omit (do not include) group (if the group is imported) when formatting the value set using `#setValue(NutsDependency)`.

```
[write-only] NutsDependencyFormat omitImportedGroup  
NutsDependencyFormat setOmitImportedGroup(omitEnv)
```

`omitImportedGroupId`

omit imported group

```
[read-only] boolean omitImportedGroupId  
boolean isOmitImportedGroupId()
```

`omitNamespace`

if true omit (do not include) namespace when formatting the value set using `#setValue(NutsDependency)`.

```
[read-write] NutsDependencyFormat omitNamespace  
boolean isOmitNamespace()  
NutsDependencyFormat setOmitNamespace(omitNamespace)
```

`omitOptional`

if true omit (do not include) face when formatting the value set using `#setValue(NutsDependency)`

```
[read-write] NutsDependencyFormat omitOptional  
boolean isOmitOptional()  
NutsDependencyFormat setOmitOptional(value)
```

`omitOtherProperties`

if true omit (do not include) query (scope and optional) when formatting the value set using `#setValue(NutsDependency)`.

```
[read-write] NutsDependencyFormat omitOtherProperties  
boolean isOmitOtherProperties()  
NutsDependencyFormat setOmitOtherProperties(value)
```

`omitQueryProperties`

list of all omitted query properties

```
[read-only] String[] omitQueryProperties  
String[] getOmitQueryProperties()
```

omitScope

if true omit (do not include) face when formatting the value set using `#setValue(NutsDependency)`

```
[read-write] NutsDependencyFormat omitScope  
boolean isOmitScope()  
NutsDependencyFormat setOmitScope(value)
```

session

update session

```
[write-only] NutsDependencyFormat session  
NutsDependencyFormat setSession(session)
```

value

value dependency to format

```
[read-write] NutsDependencyFormat value  
NutsDependency getValue()  
NutsDependencyFormat setValue(dependency)
```

Instance Methods

builder()

return mutable id builder instance initialized with `this` instance.

```
NutsDependencyBuilder builder()
```

⚙️ `configure(skipUnsupported, args)`

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsDependencyFormat configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ `isOmitQueryProperty(name)`

return true if omit query property named `name`

```
boolean isOmitQueryProperty(String name)
```

property name

⚙️ `parse(dependency)`

parse dependency in the form namespace://group:name#version?scope=<scope>` & `optional=<optional>If the string cannot be evaluated, return null.

```
NutsDependency parse(String dependency)
```

dependency

⚙️ `parseRequired(dependency)`

parse dependency in the form namespace://group:name#version?scope=<scope>` & `optional=<optional>If the string cannot be evaluated, return null.

```
NutsDependency parseRequired(String dependency)
```

dependency

⚙️ `setOmitQueryProperty(name, value)`

if true omit (do not include) query property named `name` when formatting the valueset using `#setValue(NutsDependency)` .

```
NutsDependencyFormat setOmitQueryProperty(String name, boolean value)
```

property name new value

🖨️ `NutsDescriptorFormat`

```
public net.vpc.app.nuts.NutsDescriptorFormat
```

Descriptor Format class that help building, formatting and parsing Descriptors.

🔗 Instance Properties

📝 `compact`

value compact flag. When true, formatted Descriptor will compact JSON result.

```
[read-write] NutsDescriptorFormat compact  
boolean isCompact()  
NutsDescriptorFormat setCompact(compact)
```

⚙️ Instance Methods

⚙️ `callBuilder()`

create executor builder.

```
NutsArtifactCallBuilder callBuilder()
```

⚙️ `classifierBuilder()`

create classifier mappings builder.

NutsClassifierMappingBuilder **classifierBuilder()**

⚙️ **compact()**

value compact flag to true. When true, formatted Descriptor will compact JSON result.

NutsDescriptorFormat **compact()**

⚙️ **compact(compact)**

value compact flag. When true, formatted Descriptor will compact JSON result.

NutsDescriptorFormat **compact(boolean compact)**

compact value

⚙️ **configure(skipUnsupported, args)**

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

NutsDescriptorFormat **configure(boolean skipUnsupported, String args)**

when true, all unsupported options are skipped argument to configure with

⚙️ **descriptorBuilder()**

create descriptor builder.

NutsDescriptorBuilder **descriptorBuilder()**

⚙️ **locationBuilder()**

create descriptor builder.

NutsIdLocationBuilder `locationBuilder()`

⚙️ `parse(bytes)`

parse descriptor.

NutsDescriptor `parse(byte[] bytes)`

value to parse

⚙️ `parse(descriptorString)`

parse descriptor.

NutsDescriptor `parse(String descriptorString)`

string to parse

⚙️ `parse(file)`

parse descriptor.

NutsDescriptor `parse(File file)`

file to parse

⚙️ `parse(path)`

parse descriptor.

NutsDescriptor `parse(Path path)`

path to parse

⚙️ parse(stream)

parse descriptor.

```
NutsDescriptor parse(InputStream stream)
```

stream to parse

⚙️ parse(url)

parse descriptor.

```
NutsDescriptor parse(URL url)
```

URL to parse

⚙️ value(descriptor)

set the descriptor instance to print

```
NutsDescriptorFormat value(NutsDescriptor descriptor)
```

value to format

🖨️ NutsElementFormat

```
public net.vpc.app.nuts.NutsElementFormat
```

Class responsible of manipulating `NutsElement` type. It help parsing from, converting to and formatting such types.

🔗 Instance Properties

⌚ session

set current session.

```
[write-only] NutsElementFormat session  
NutsElementFormat setSession(session)
```

value

set current value to format.

```
[read-write] NutsElementFormat value  
Object getValue()  
NutsElementFormat setValue(value)
```

Instance Methods

builder()

element builder

```
NutsElementBuilder builder()
```

compilePath(pathExpression)

compile pathExpression into a valid NutsElementPath that helps filtering elements tree. JSONPath expressions refer to a JSON structure the same way as XPath expression are used with XML documents. JSONPath expressions can use the dot notation and/or bracket notations .store.book[0].title The trailing root is not necessary : .store.book[0].title You can also use bracket notation store['book'][0].title for input paths.

```
NutsElementPath compilePath(String pathExpression)
```

element path expression

configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)`

to help return a more specific return type;

NutsElementFormat configure(boolean skipUnsupported, String args)

when true, all unsupported options are skipped
argument to configure with

⚙ fromElement(element, clazz)
convert element to the specified object if applicable or throw an exception.

T fromElement(NutsElement element, Class<T> clazz)

element to convert
class type

⚙ set(value)
set current value to format.

NutsElementFormat set(Object value)

value to format

⚙ toElement(object)
convert any object to valid `NutsElement`.

NutsElement toElement(Object object)

object to convert

☕ NutsExecCommandFormat

public net.vpc.app.nuts.NutsExecCommandFormat

Format used to format command line by ``` NutsExecCommand````

Instance Properties

argumentFilter
set arg filter

[read-write] NutsExecCommandFormat argumentFilter Predicate\<ArgEntry\> getArgumentFilter()
NutsExecCommandFormat setArgumentFilter(filter)

argumentReplacer
set arg replacer

[read-write] NutsExecCommandFormat argumentReplacer Function\<ArgEntry,String\>
getArgumentReplacer() NutsExecCommandFormat setArgumentReplacer(replacer)

envFilter
set env filter

[read-write] NutsExecCommandFormat envFilter Predicate\<EnvEntry\> getEnvFilter()
NutsExecCommandFormat setEnvFilter(filter)

envReplacer
set env replacer

[read-write] NutsExecCommandFormat envReplacer Function\<EnvEntry,String\> getEnvReplacer()
NutsExecCommandFormat setEnvReplacer(replacer)

redirectError
if true error redirection is displayed

[read-write] NutsExecCommandFormat redirectError boolean isRedirectError()
NutsExecCommandFormat setRedirectError(redirectError)

redirectInput
if true input redirection is displayed

[read-write] NutsExecCommandFormat redirectInput boolean isRedirectInput()

NutsExecCommandFormat setRedirectInput(redirectInput)

redirectOutput
if true output redirection is displayed

[read-write[NutsExecCommandFormat redirectOutput boolean isRedirectOutput()
NutsExecCommandFormat setRedirectOutput(redirectOutput)

value
set value to format

[read-write[NutsExecCommandFormat value NutsExecCommand getValue()
NutsExecCommandFormat setValue(value)

Instance Methods
value(value)
set value to format

NutsExecCommandFormat value(NutsExecCommand value)

value to format
NutsFormat

public net.vpc.app.nuts.NutsFormat

Base Format Interface used to print "things".
Instance Properties
session
update session

[read-write[NutsFormat session NutsSession getSession() NutsFormat setSession(session)

Instance Methods
configure(skipUnsupported, args)
configure the current command with the given arguments. This is an
override of the `` NutsConfigurable#configure(boolean, java.lang.String...)

to help return a more specific return type;

```
NutsFormat configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

 **format()**

format current value and return the string result

```
String format()
```

 **print()**

format current value and write result to ` getSession().out()` .

```
void print()
```

 **print(out)**

format current value and write result to ` out`

```
void print(PrintStream out)
```

recipient print stream

 **print(out)**

format current value and write result to ` out`

```
void print(Writer out)
```

recipient writer

⚙️ print(out)

format current value and write result to `out`

```
void print(OutputStream out)
```

recipient writer

⚙️ print(out)

format current value and write result to `out`

```
void print(Path out)
```

recipient path

⚙️ print(out)

format current value and write result to `out`

```
void print(File out)
```

recipient file

⚙️ print(terminal)

format current value and write result to `terminal`

```
void print(NutsTerminal terminal)
```

recipient terminal

⚙️ println()

format current value and write result to `getSession().out()` and finally appends a new line.

```
void println()
```

⚙ **println(file)**

format current value and write result to `out` and finally appends a new line.

```
void println(File file)
```

recipient file

⚙ **println(out)**

format current value and write result to `out` and finally appends a new line.

```
void println(Writer out)
```

recipient

⚙ **println(out)**

format current value and write result to `out` and finally appends a new line.

```
void println(PrintStream out)
```

recipient print stream

⚙ **println(out)**

format current value and write result to `out` and finally appends a new line.

```
void println(Path out)
```

recipient path

`println(terminal)`

format current value and write result to ` terminal` and finally appends a new line.

```
void println(NutsTerminal terminal)
```

recipient terminal

`toString()`

equivalent to ` #format() `

```
String toString()
```

NutsIdFormat

```
public net.vpc.app.nuts.NutsIdFormat
```

Class responsible of manipulating ` NutsId` instances:

- formatting (in Nuts Stream Format)
- parsing

`highlightImportedGroupId`

update `highlightImportedGroupId`

```
[read-write] NutsIdFormat highlightImportedGroupId
boolean isHighlightImportedGroupId()
NutsIdFormat setHighlightImportedGroupId(value)
```

`highlightOptional`

update `highlightOptional`

```
[read-write] NutsIdFormat highlightOptional  
boolean isHighlightOptional()  
NutsIdFormat setHighlightOptional(value)
```

highlightScope

update highlightScope

```
[read-write] NutsIdFormat highlightScope  
boolean isHighlightScope()  
NutsIdFormat setHighlightScope(value)
```

omitClassifier

if true omit (do not include) face when formatting the value set using `#setValue(NutsId)` .

```
[read-write] NutsIdFormat omitClassifier  
boolean isOmitClassifier()  
NutsIdFormat setOmitClassifier(value)
```

omitFace

update omitFace

```
[read-write] NutsIdFormat omitFace  
boolean isOmitFace()  
NutsIdFormat setOmitFace(value)
```

omitGroupId

update omitGroup

```
[read-write] NutsIdFormat omitGroupId  
boolean isOmitGroupId()  
NutsIdFormat setOmitGroupId(value)
```

`omitImportedGroupId`

update `omitImportedGroupId`

```
[read-write] NutsIdFormat omitImportedGroupId  
boolean isOmitImportedGroupId()  
NutsIdFormat setOmitImportedGroupId(value)
```

`omitNamespace`

update `omitNamespace`

```
[read-write] NutsIdFormat omitNamespace  
boolean isOmitNamespace()  
NutsIdFormat setOmitNamespace(value)
```

`omitOtherProperties`

update `omitOtherProperties`

```
[read-write] NutsIdFormat omitOtherProperties  
boolean isOmitOtherProperties()  
NutsIdFormat setOmitOtherProperties(value)
```

`omitProperties`

query properties omitted

```
[read-only] String[] omitProperties  
String[] getOmitProperties()
```

`session`

update `session`

```
[write-only] NutsIdFormat session  
NutsIdFormat setSession(session)
```

value

id to format

```
[read-write] NutsIdFormat value  
NutsId getValue()  
NutsIdFormat setValue(id)
```

Instance Methods

builder()

create new instance of id builder

```
NutsIdBuilder builder()
```

configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsIdFormat configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

highlightImportedGroupId()

update highlightImportedGroupId to true

```
NutsIdFormat highlightImportedGroupId()
```

⚙️ **highlightImportedGroupId(value)**

update highlightImportedGroupId

NutsIdFormat **highlightImportedGroupId(boolean value)**

value

⚙️ **highlightOptional()**

update highlightOptional tot true

NutsIdFormat **highlightOptional()**

⚙️ **highlightOptional(value)**

update highlightOptional

NutsIdFormat **highlightOptional(boolean value)**

value

⚙️ **highlightScope()**

update highlightScope to true

NutsIdFormat **highlightScope()**

⚙️ **highlightScope(value)**

update highlightScope

NutsIdFormat **highlightScope(boolean value)**

value

⚙️ **isOmitProperty(name)**

return true if omit query property named `name`

```
boolean isOmitProperty(String name)
```

property name

⚙️ **omitClassifier()**

omit scope

```
NutsIdFormat omitClassifier()
```

⚙️ **omitClassifier(value)**

if true omit (do not include) face when formatting the value set using `#setValue(NutsId)` .

```
NutsIdFormat omitClassifier(boolean value)
```

new value

⚙️ **omitFace()**

update omitFace to true

```
NutsIdFormat omitFace()
```

⚙️ **omitFace(value)**

update omitFace

```
NutsIdFormat omitFace(boolean value)
```

value

⚙️ `omitGroupId()`

update omitGroup to true

`NutsIdFormat omitGroupId()`

⚙️ `omitGroupId(value)`

update omitGroup

`NutsIdFormat omitGroupId(boolean value)`

new value

⚙️ `omitImportedGroupId()`

update omitImportedGroupId to true

`NutsIdFormat omitImportedGroupId()`

⚙️ `omitImportedGroupId(value)`

update omitImportedGroupId

`NutsIdFormat omitImportedGroupId(boolean value)`

value

⚙️ `omitNamespace()`

update omitNamespace to true

`NutsIdFormat omitNamespace()`

⚙️ **omitNamespace(value)**

update omitNamespace

```
NutsIdFormat omitNamespace(boolean value)
```

true when the namespace should not be included in formatted instance

⚙️ **omitOtherProperties()**

update omitOtherProperties to true

```
NutsIdFormat omitOtherProperties()
```

⚙️ **omitOtherProperties(value)**

update omitOtherProperties

```
NutsIdFormat omitOtherProperties(boolean value)
```

value

⚙️ **omitProperty(name)**

omit query property named `name`

```
NutsIdFormat omitProperty(String name)
```

property name

⚙️ **omitProperty(name, value)**

if true omit (do not include) query property named `name` when formatting the valueset using `#setValue(NutsId)` .

```
NutsIdFormat omitProperty(String name, boolean value)
```

property name new value

⚙️ **parse(id)**

parse id or null if not valid. id is parsed in the form namespace://group:name#version?key=<value>
` & ` key=<value> ...

NutsId **parse(String id)**

to parse

⚙️ **parseRequired(id)**

parse id or error if not valid

NutsId **parseRequired(String id)**

to parse

⚙️ **resolveld(clazz)**

detect nuts id from resources containing the given class or null if not found. If multiple resolutions return the first.

NutsId **resolveId(Class clazz)**

to search for

⚙️ **resolvelds(clazz)**

detect all nuts ids from resources containing the given class.

NutsId[] **resolveIds(Class clazz)**

to search for

⚙️ set(id)

id to format

```
NutsIdFormat set(NutsId id)
```

id to format

⚙️ setOmitProperty(name, value)

if true omit (do not include) query property named `name` when formatting the valueset using `#setValue(NutsId)` .

```
NutsIdFormat setOmitProperty(String name, boolean value)
```

property name new value

⚙️ value(id)

set id to format

```
NutsIdFormat value(NutsId id)
```

id to format

🖨️ NutsIterableFormat

```
public net.vpc.app.nuts.NutsIterableFormat
```

This class handles formatting of iterable items in Search.

🔗 Instance Properties

📄 ↴ outputFormat

Current output format

```
[read-only] NutsOutputFormat outputFormat  
NutsOutputFormat getOutputFormat()
```

⚙ Instance Methods

⚙ complete(count)

called at the iteration completing

```
void complete(long count)
```

iterated items count

⚙ configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsIterableFormat configure(boolean skipUnsupported, String args)
```

null argument to configure with

⚙ next(object, index)

called at each new item visited

```
void next(Object object, long index)
```

visited item visited item index

⚙ start()

called at the iteration start

```
void start()
```

☕ NutsIterableOutput

```
public net.vpc.app.nuts.NutsIterableOutput
```

Classes implementing this interface are responsible of printing objects in multiple format using `NutsIterableFormat` .TODO : should merge with NutsIterableFormat

⌚ Instance Properties

⌚ out

configure out c

```
[write-only] NutsIterableOutput out  
NutsIterableOutput setOut(out)
```

⌚ session

configure session

```
[write-only] NutsIterableOutput session  
NutsIterableOutput setSession(session)
```

⚙️ Instance Methods

⚙️ complete()

called at the iteration completing

```
void complete()
```

⚙️ configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsIterableOutput configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

 **next(object)**

called at the each visited item

```
void next(Object object)
```

visited item

 **out(out)**

configure out stream

```
NutsIterableOutput out(PrintStream out)
```

out stream

 **out(out)**

configure out stream

```
NutsIterableOutput out(PrintWriter out)
```

out stream

 **start()**

called at the iteration start

```
void start()
```

☕ NutsJsonFormat

```
public net.vpc.app.nuts.NutsJsonFormat
```

Implementation of this interface will provide simple mechanism to write json text from given object.

⌚ Instance Properties

📝⌚ compact

enable or disable compact json

```
[read-write] NutsJsonFormat compact  
boolean isCompact()  
NutsJsonFormat setCompact(compact)
```

⌚⌚ session

update session

```
[write-only] NutsJsonFormat session  
NutsJsonFormat setSession(session)
```

📝⌚ value

```
[read-write] NutsJsonFormat value  
Object getValue()  
NutsJsonFormat setValue(value)
```

⚙️ Instance Methods

⚙️ compact()

enable compact json

NutsJsonFormat compact()**⚙️ compact(compact)**

enable or disable compact json

NutsJsonFormat compact(boolean compact)

enable when true

⚙️ configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)`

to help return a more specific return type;

NutsJsonFormat configure(boolean skipUnsupported, String args)

when true, all unsupported options are skipped
argument to configure with

⚙️ parse(bytes, clazz)
parse bytes as a valid object of the given type

T parse(byte[] bytes, Class<T> clazz)

source bytes
target type

⚙️ parse(file, clazz)
parse file as a valid object of the given type

T parse(Path file, Class<T> clazz)

```
source url  
target type  
  
#### ⚭ parse(file, clazz)  
parse file as a valid object of the given type
```

T parse(File file, Class<T> clazz)

```
source url  
target type  
  
#### ⚭ parse(inputStream, clazz)  
parse inputStream as a valid object of the given type
```

T parse(InputStream inputStream, Class<T> clazz)

```
source inputStream  
target type  
  
#### ⚭ parse(jsonString, clazz)  
parse inputStream as a valid object of the given type
```

T parse(String jsonString, Class<T> clazz)

```
source as json string  
target type  
  
#### ⚭ parse(reader, clazz)  
parse reader as a valid object of the given type
```

T parse(Reader reader, Class<T> clazz)

```
source reader  
target type  
  
#### ⚭ parse(url, clazz)  
parse url as a valid object of the given type
```

T parse(URL url, Class<T> clazz)

```
source url  
target type  
  
#### ⚙ value(value)
```

NutsJsonFormat value(Object value)

```
value to format  
  
## ☕ NutsMutableTableModel
```

public net.vpc.app.nuts.NutsMutableTableModel

```
Mutable Table Model  
### ⚙ Instance Methods  
#### ⚙ addCell(value)  
add row cell
```

NutsMutableTableModel addCell(Object value)

```
cell  
  
#### ⚙ addCells(values)  
add row cells
```

NutsMutableTableModel addCells(Object values)

```
row cells  
  
#### ⚙ addHeaderCell(value)  
add header cell
```

NutsMutableTableModel addHeaderCell(Object value)

cell

```
#### ⚙ addHeaderCells(values)  
add header cells
```

NutsMutableTableModel addHeaderCells(Object values)

cells

```
#### ⚙ addRow(values)  
add row cells
```

NutsMutableTableModel addRow(Object values)

row cells

```
#### ⚙ clearHeader()  
clear header
```

NutsMutableTableModel clearHeader()

```
#### ⚙ newRow()  
add new row to the model
```

NutsMutableTableModel newRow()

```
#### ⚙ setCellColSpan(row, column, value)  
update cell colspan
```

NutsMutableTableModel setCellColSpan(int row, int column, int value)

row index
column index
new value

```
#### ⚙ setCellRowSpan(row, column, value)  
update cell rowspan
```

NutsMutableTableModel setCellRowSpan(int row, int column, int value)

```
row index
column index
new value

#### ⚙️ setCellValue(row, column, value)
update cell at the given position
```

NutsMutableTableModel setCellValue(int row, int column, Object value)

```
row index
column index
cell value

#### ⚙️ setHeaderColSpan(column, value)
update header colspan
```

NutsMutableTableModel setHeaderColSpan(int column, int value)

```
new value
new value

#### ⚙️ setHeaderValue(column, value)
update header value
```

NutsMutableTableModel setHeaderValue(int column, Object value)

```
header column
new value

## 🎉 NutsNamedElement
```

public net.vpc.app.nuts.NutsNamedElement

```
Named Element
### 📂 Instance Properties
#### 📄 name
element name
```

[read-only] String name String getName()

```
#### 📄 value  
element value
```

[read-only] NutsElement value NutsElement getValue()

```
## 🎉 NutsObjectElementBuilder
```

public net.vpc.app.nuts.NutsObjectElementBuilder

```
Builder for manipulating ``` NutsObjectElement``` instances
```

```
### ⚙ Instance Methods
```

```
#### ⚙ add(other)
```

```
set all properties from the given ``` other``` instance.
```

```
all properties not found in ``` other``` will be retained.
```

NutsObjectElementBuilder add(NutsObjectElement other)

```
other instance
```

```
#### ⚙ add(other)
```

```
set all properties from the given ``` other``` instance.
```

```
all properties not found in ``` other``` will be retained.
```

NutsObjectElementBuilder add(NutsObjectElementBuilder other)

```
other instance
```

```
#### ⚙ build()
```

```
create a immutable instance of ``` NutsObjectElement``` representing  
this builder.
```

NutsObjectElement build()

```
#### ⚙ children()
```

```
object (key,value) attributes
```

Collection\<NutsNamedElement\> children()

```
#### ⚙ clear()
remove all properties
```

NutsObjectElementBuilder clear()

```
#### ⚙ get(name)
return value for name or null.
If multiple values are available return any of them.
```

NutsElement get(String name)

```
key name

#### ⚙ remove(name)
remove property
```

NutsObjectElementBuilder remove(String name)

```
property name

#### ⚙ set(other)
set all properties from the given `other` instance.
all properties not found in `other` will be removed.
```

NutsObjectElementBuilder set(NutsObjectElement other)

```
other instance

#### ⚙ set(other)
set all properties from the given `other` instance.
all properties not found in `other` will be removed.
```

NutsObjectElementBuilder set(NutsObjectElementBuilder other)

other instance

```
#### ⚙ set(name, value)  
set value for property `name`
```

NutsObjectElementBuilder set(String name, NutsElement value)

property name
property value. should not be null

```
#### ⚙ size()  
element count
```

int size()

```
## 📁 NutsObjectFormat
```

public net.vpc.app.nuts.NutsObjectFormat

Object format is responsible of formatting to terminal
a given object. Multiple implementation should be available
to support tables, trees, json, xml,...

```
### 🌐 Instance Properties  
### 🌐 session  
update session
```

[write-only] NutsObjectFormat session NutsObjectFormat setSession(session)

```
#### 🖊️ 🌐 value  
set value to format
```

[read-write] NutsObjectFormat value Object getValue() NutsObjectFormat setValue(value)

```
### ⚙ Instance Methods
#### ⚙ configure(skipUnsupported, args)
configure the current command with the given arguments. This is an
override of the `` NutsConfigurable#configure(boolean, java.lang.String...) `````` to help return a more specific return type;
```

NutsObjectFormat configure(boolean skipUnsupported, String args)

when true, all unsupported options are skipped
argument to configure with

```
#### ⚙ value(value)
set value to format
```

NutsObjectFormat value(Object value)

value to format

```
## 📄 NutsPropertiesFormat
```

public net.vpc.app.nuts.NutsPropertiesFormat

Class formatting Map/Properties objects
☰ Instance Properties
🖊️ ☰ model
set model to format

[read-write] NutsPropertiesFormat model Map getModel() NutsPropertiesFormat setModel(map)

```
#### 🖊️ ☰ separator
set key/value separator
```

[read-write] NutsPropertiesFormat separator String getSeparator() NutsPropertiesFormat setSeparator(separator)

```
#### ☰ session
update session
```

[write-only] NutsPropertiesFormat session NutsPropertiesFormat setSession(session)

```
#### 🖊 sort  
enable/disable key sorting
```

[read-write] NutsPropertiesFormat sort boolean isSort() NutsPropertiesFormat setSort(sort)

```
### ⚙ Instance Methods  
#### ⚙ configure(skipUnsupported, args)  
configure the current command with the given arguments. This is an  
override of the `` NutsConfigurable#configure(boolean, java.lang.String...)
```

to help return a more specific return type;

```
NutsPropertiesFormat configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙ model(map)

set model to format

```
NutsPropertiesFormat model(Map map)
```

model to format

⚙ separator(separator)

set key/value separator

```
NutsPropertiesFormat separator(String separator)
```

key/value separator

⚙ sort()

enable key sorting

NutsPropertiesFormat `sort()`

⚙️ `sort(boolean sort)`

enable/disable key sorting

NutsPropertiesFormat `sort(boolean sort)`

when true enable sorting

☕ NutsQuestionFormat

public net.vpc.app.nuts.NutsQuestionFormat

⚙️ Instance Methods

⚙️ `format(value, question)`

String `format(Object value, NutsQuestion<T> question)`

null null

⚙️ `getDefaultValues(type, question)`

Object[] `getDefaultValues(Class type, NutsQuestion<T> question)`

null null

☕ NutsString

public net.vpc.app.nuts.NutsString

Constructors

NutsString(value)

```
NutsString(String value)
```

null

Instance Properties

value

```
[read-only] public String value  
private String value  
public String getValue()
```

Instance Methods

equals(o)

```
boolean equals(Object o)
```

null

hashCode()

```
int hashCode()
```

toString()

```
String toString()
```

☕ NutsStringFormat

```
public net.vpc.app.nuts.NutsStringFormat
```

Class responsible of formatting a formatted string.

Instance Properties

📝 parameters

```
[read-write] NutsStringFormat parameters  
Object[] getParameters()  
NutsStringFormat setParameters(parameters)
```

session

set current session.

```
[write-only] NutsStringFormat session  
NutsStringFormat setSession(session)
```

📝 string

set current value to format.

```
[read-write] NutsStringFormat string  
String getString()  
NutsStringFormat setString(value)
```

📝 style

```
[read-write] NutsStringFormat style  
NutsTextFormatStyle getStyle()  
NutsStringFormat setStyle(style)
```

⚙ Instance Methods

⚙ addParameters(parameters)

```
NutsStringFormat addParameters(Object parameters)
```

null

⚙ append(value, parameters)

```
NutsStringFormat append(String value, Object parameters)
```

null null

⚙ configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, String...)`

```
to help return a more specific return type;
```

NutsStringFormat configure(boolean skipUnsupported, String args)

```
when true, all unsupported options are skipped  
argument to configure with
```

```
#### ⚙ of(value, parameters)
```

NutsStringFormat of(String value, Object parameters)

```
null  
null
```

```
#### ⚙ set(value)  
set current value to format.
```

NutsStringFormat set(String value)

value to format

⚙️ style(NutsTextStyle style)

NutsStringFormat style(NutsTextStyle style)

null

☕ NutsTableBordersFormat

public net.vpc.app.nuts.NutsTableBordersFormat

⚙️ Instance Methods

⚙️ format(s)

String format(Separator s)

null

☕ NutsTableCell

public net.vpc.app.nuts.NutsTableCell

📊 Instance Properties

🖍️ colspans

[read-write] NutsTableCell colspans int getColspan() NutsTableCell setColspan(colspan)

🖍️ rowspan

[read-write] NutsTableCell rowspans int getRowspan() NutsTableCell setRowspan(rowspan)

🖍️ value

[read-write[NutsTableCell value Object getValue() NutsTableCell setValue(value)

 x

[read-only[int x int getX()

 y

[read-only[int y int getY()

 NutsTableCellFormat

public net.vpc.app.nuts.NutsTableCellFormat

 Instance Methods

 format(row, col, value)

String format(int row, int col, Object value)

null

null

null

 getHorizontalAlign(row, col, value)

NutsPositionType getHorizontalAlign(int row, int col, Object value)

null

null

null

 getVerticalAlign(row, col, value)

NutsPositionType getVerticalAlign(int row, int col, Object value)

```
null  
null  
null  
  
## ☕ NutsTableFormat
```

```
public net.vpc.app.nuts.NutsTableFormat
```

```
### ☐ Instance Properties  
#### 📝 ☐ border
```

```
[read-write[ NutsTableFormat border NutsTableBordersFormat getBorder() NutsTableFormat  
setBorder(border)
```

```
#### ☐ cellFormat
```

```
[write-only[ NutsTableFormat cellFormat NutsTableFormat setCellFormat(formatter)
```

```
#### 📝 ☐ model
```

```
[read-write[ NutsTableFormat model NutsTableModel getModel() NutsTableFormat setModel(model)
```

```
#### ☐ session  
update session
```

```
[write-only[ NutsTableFormat session NutsTableFormat setSession(session)
```

```
#### 📝 ☐ visibleHeader
```

```
[read-write[ NutsTableFormat visibleHeader boolean isVisibleHeader() NutsTableFormat  
setVisibleHeader(visibleHeader)
```

```
### ⚙ Instance Methods
#### ⚙ configure(skipUnsupported, args)
configure the current command with the given arguments. This is an
override of the ``` NutsConfigurable#configure(boolean, java.lang.String...) ``` 
to help return a more specific return type;
```

NutsTableFormat configure(boolean skipUnsupported, String args)

```
when true, all unsupported options are skipped
argument to configure with
```

```
#### ⚙ createModel()
```

NutsMutableTableModel createModel()

```
#### ⚙ getVisibleColumn(col)
```

Boolean getVisibleColumn(int col)

```
null
```

```
#### ⚙ setVisibleColumn(col, visible)
```

NutsTableFormat setVisibleColumn(int col, boolean visible)

```
null
```

```
null
```

```
#### ⚙ unsetVisibleColumn(col)
```

NutsTableFormat unsetVisibleColumn(int col)

```
null
```

```
## ☕ NutsTableModel
```

public net.vpc.app.nuts.NutsTableModel

```
### □ Instance Properties  
#### 📋□ columnsCount
```

[read-only] int columnsCount int getColumnsCount()

```
#### 📋□ rowsCount
```

[read-only] int rowsCount int getRowsCount()

```
### ⚙ Instance Methods  
#### ⚙ getCellColSpan(row, column)
```

int getCellColSpan(int row, int column)

```
null  
null  
  
#### ⚙ getCellRowSpan(row, column)
```

int getCellRowSpan(int row, int column)

```
null  
null  
  
#### ⚙ getCellValue(row, column)
```

Object getCellValue(int row, int column)

```
null  
null  
  
#### ⚙ getHeaderColSpan(column)
```

int getHeaderColSpan(int column)

```
null  
#### ⚙ getHeaderValue(column)
```

Object getHeaderValue(int column)

```
null  
## 🎣 NutsTerminalFormat
```

public net.vpc.app.nuts.NutsTerminalFormat

```
Filtered Terminal Format Helper  
### ⚙ Instance Methods  
#### ⚙ escapeText(value)  
This method escapes all special characters that are interpreted by  
"nuts print format" so that this exact string is printed on  
such print streams When str is null, an empty string is return
```

String escapeText(String value)

```
input string  
#### ⚙ filterText(value)  
this method removes all special "nuts print format" sequences support  
and returns the raw string to be printed on an  
ordinary `PrintStream`
```

String filterText(String value)

```
input string  
#### ⚙ formatText(style, format, args)  
format string. supports `Formatter#format(java.lang.String,  
java.lang.Object...)`
```

pattern format and adds NutsString special format to print unfiltered strings.

```
String formatText(NutsTextStyle style, String format, Object args)
```

format style format arguments

⚙️ formatText(style, locale, format, args)

format string. supports `Formatter#format(java.util.Locale, java.lang.String, java.lang.Object...)`

pattern format and adds NutsString special format to print unfiltered strings.

String formatText(NutsTextStyle style, Locale locale, String format, Object args)

style
locale
format
arguments

⚙️ isFormatted(out)
true if the stream is not null and could be resolved as Formatted Output Stream. If False is returned this does no mean necessarily that the stream is not formatted.

boolean isFormatted(OutputStream out)

stream to check

⚙️ isFormatted(out)
true if the stream is not null and could be resolved as Formatted Output Stream. If False is returned this does no mean necessarily that the stream is not formatted.

boolean isFormatted(Writer out)

stream to check

⚙️ prepare(out)
prepare PrintStream to handle NutsString aware format pattern. If the instance already supports Nuts specific pattern it will be returned unmodified.

PrintStream prepare(PrintStream out)

PrintStream to check

⚙️ prepare(out)

prepare PrintWriter to handle %N (escape) format pattern. If the instance already supports Nuts specific pattern it will be returned unmodified.

PrintWriter prepare(PrintWriter out)

PrintWriter to check

⚙️ textLength(value)

int textLength(String value)

null

📄 NutsTreeFormat

public net.vpc.app.nuts.NutsTreeFormat

Tree Format handles terminal output in Tree format.

It is one of the many formats supported by nuts such as plain, table, xml, json.

To use Tree format, given an instance ws of Nuts Workspace you can :

ws.

📄 Instance Properties

🖊️📄 linkFormat

update link format

[read-write[NutsTreeFormat linkFormat NutsTreeLinkFormat getLinkFormat() NutsTreeFormat
setLinkFormat(linkFormat)

```
#### 🖊 model
update tree model
```

[read-write[NutsTreeFormat model NutsTreeModel getModel() NutsTreeFormat setModel(tree)

```
#### 🖊 nodeFormat
update node format
```

[read-write[NutsTreeFormat nodeFormat NutsTreeNodeFormat getNodeFormat() NutsTreeFormat setNodeFormat(nodeFormat)

```
#### 🚦 session
update session
```

[write-only[NutsTreeFormat session NutsTreeFormat setSession(session)

```
### ⚙ Instance Methods
#### ⚙ configure(skipUnsupported, args)
configure the current command with the given arguments. This is an
override of the ``` NutsConfigurable#configure(boolean, java.lang.String...) ```` to help return a more specific return type;
```

NutsTreeFormat configure(boolean skipUnsupported, String args)

when true, all unsupported options are skipped
argument to configure with

```
#### ⚙ linkFormat(linkFormat)
update link format
```

NutsTreeFormat linkFormat(NutsTreeLinkFormat linkFormat)

new link format

```
#### ⚙ model(tree)
update tree model
```

NutsTreeFormat model(NutsTreeModel tree)

```
new tree model

#### ⚙ nodeFormat(nodeFormat)
update node format
```

NutsTreeFormat nodeFormat(NutsTreeNodeFormat nodeFormat)

```
new node format

## ☕ NutsTreeLinkFormat
```

public net.vpc.app.nuts.NutsTreeLinkFormat

```
Format class responsible of formatting prefix of a tree
### ⚙ Instance Methods
#### ⚙ formatChild(type)
return prefix for node child for the given layout
```

String formatChild(NutsPositionType type)

```
position type

#### ⚙ formatMain(type)
return prefix for node root for the given layout
```

String formatMain(NutsPositionType type)

```
position type

## ☕ NutsTreeModel
```

public net.vpc.app.nuts.NutsTreeModel

Tree Model to use in tree format
 ### ☈ Instance Properties
 ##### 📄 ☈ root
 tree node

[read-only] Object root Object getRoot()

☈ Instance Methods
 ##### ☈ getChildren(node)
 return children of the given `node`

List<T> getChildren(Object node)

node to retrieve children for
 ## ☕ NutsTreeNodeFormat

public net.vpc.app.nuts.NutsTreeNodeFormat

classes implementing this interface handle formatting of the tree node.
 ### ☈ Instance Methods
 ##### ☈ format(object, depth)
 format (transform to rich string) object at the given depth

String format(Object object, int depth)

object to transform
 tree node depth
 ## ☕ NutsVersionFormat

public net.vpc.app.nuts.NutsVersionFormat

☈ Instance Properties
 ##### ☈ session
 update session

[write-only] NutsVersionFormat session NutsVersionFormat setSession(session)

version
set version to print. if null, workspace version will be considered.

[read-write] NutsVersionFormat version NutsVersion getVersion() NutsVersionFormat setVersion(version)

workspaceVersion
return true if version is null (default). In such case, workspace version is considered.

[read-only] boolean workspaceVersion boolean isWorkspaceVersion()

Instance Methods
addProperties(p)

NutsVersionFormat addProperties(Map<String, String> p)

null
addProperty(key, value)

NutsVersionFormat addProperty(String key, String value)

null
null
parse(version)
return version instance representing the ```version``` string

NutsVersion parse(String version)

string (may be null)
NutsWorkspaceOptionsFormat

```
public net.vpc.app.nuts.NutsWorkspaceOptionsFormat
```

```
    ### ☈ Constant Fields  
    ##### ☈ serialVersionUID
```

```
private static final long serialVersionUID = 1
```

```
    ### ☈ Instance Fields  
    ##### ☈ createOptions
```

```
private boolean createOptions
```

```
    ##### ☈ exportedOptions
```

```
private boolean exportedOptions
```

```
    ##### ☈ omitDefaults
```

```
private boolean omitDefaults
```

```
    ##### ☈ options
```

```
private final NutsWorkspaceOptions options
```

```
    ##### ☈ runtimeOptions
```

```
private boolean runtimeOptions
```

```
    ##### ☈ shortOptions
```

```
private boolean shortOptions
```

```
    ##### ☈ singleArgOptions
```

```
private boolean singleArgOptions
```

```
### □ Constructors
#### □ NutsWorkspaceOptionsFormat(options)
```

NutsWorkspaceOptionsFormat(NutsWorkspaceOptions options)

null

```
### □ Instance Properties
#### 📄 □ bootCommand
```

[read-only] public String[] bootCommandpublic String[] getBootCommand()

```
#### 📄 □ bootCommandLine
```

[read-only] public String bootCommandLinepublic String getBootCommandLine()

```
#### □ compact
```

[write-only] NutsWorkspaceOptionsFormat public compactpublic NutsWorkspaceOptionsFormat setCompact(compact)

```
#### 📄 □ exported
```

[read-write] NutsWorkspaceOptionsFormat public exportedpublic boolean isExported() public NutsWorkspaceOptionsFormat setExported(e)

```
#### 📄 □ implicitAll
```

[read-only] private boolean implicitAllprivate boolean isImplicitAll()

```
#### 📄 □ init
```

[read-write] NutsWorkspaceOptionsFormat public initpublic boolean isInit() public NutsWorkspaceOptionsFormat setInit(e)

```
#### 📄 runtime
```

```
[read-write] NutsWorkspaceOptionsFormat public runtimepublic boolean isRuntime() public  
NutsWorkspaceOptionsFormat setRuntime(e)
```

```
### ⚙ Instance Methods  
#### ⚙ compact()
```

```
NutsWorkspaceOptionsFormat compact()
```

```
#### ⚙ compact(compact)
```

```
NutsWorkspaceOptionsFormat compact(boolean compact)
```

```
null
```

```
#### ⚙ equals(o)
```

```
boolean equals(Object o)
```

```
null
```

```
#### ⚙ exported()
```

```
NutsWorkspaceOptionsFormat exported()
```

```
#### ⚙ exported(e)
```

```
NutsWorkspaceOptionsFormat exported(boolean e)
```

```
null
```

```
#### ⚙ fillOption(value, arguments, forceSingle)
```

```
void fillOption(Enum value, List<String> arguments, boolean forceSingle)
```

```
null  
null  
null
```

```
#### ⚙ fillOption(longName, shortName, value, arguments, forceSingle)
```

```
void fillOption(String longName, String shortName, char[] value, List<String> arguments, boolean  
forceSingle)
```

```
null  
null  
null  
null  
null
```

```
#### ⚙ fillOption(longName, shortName, value, arguments, forceSingle)
```

```
void fillOption(String longName, String shortName, String value, List<String> arguments, boolean  
forceSingle)
```

```
null  
null  
null  
null  
null
```

```
#### ⚙ fillOption(longName, shortName, value, arguments, forceSingle)
```

```
void fillOption(String longName, String shortName, int value, List<String> arguments, boolean  
forceSingle)
```

```
null  
null  
null  
null  
null
```

```
#### ⚙ fillOption(longName, shortName, value, defaultValue, arguments,  
forceSingle)
```

```
void fillOption(String longName, String shortName, boolean value, boolean defaultValue,  
List<String> arguments, boolean forceSingle)
```

```
null  
null  
null  
null  
null  
null
```

```
#### ⚙ fillOption(longName, shortName, value, enumType, arguments, forceSingle)
```

```
void fillOption(String longName, String shortName, Enum value, Class enumType, List<String>  
arguments, boolean forceSingle)
```

```
null  
null  
null  
null  
null  
null
```

```
#### ⚙ fillOption(longName, shortName, values, sep, arguments, forceSingle)
```

```
void fillOption(String longName, String shortName, String[] values, String sep, List<String>  
arguments, boolean forceSingle)
```

```
null  
null  
null  
null  
null  
null
```

```
#### ⚙ fillOption0(name, value, arguments, forceSingle)
```

```
void fillOption0(String name, String value, List<String> arguments, boolean forceSingle)
```

```
null  
null  
null  
null  
#### ⚙ hashCode()
```

int hashCode()

```
#### ⚙ init()
```

NutsWorkspaceOptionsFormat init()

```
#### ⚙ init(e)
```

NutsWorkspaceOptionsFormat init(boolean e)

```
null  
#### ⚙ runtime()
```

NutsWorkspaceOptionsFormat runtime()

```
#### ⚙ runtime(e)
```

NutsWorkspaceOptionsFormat runtime(boolean e)

```
null  
#### ⚙ selectOptionName(longName, shortName)
```

String selectOptionName(String longName, String shortName)

```
null
null

#### ⚙ selectOptionVal(shortName, longName)
```

String selectOptionVal(String shortName, String longName)

```
null
null

#### ⚙ toString()
```

String toString()

```
#### ⚙ tryFillOptionShort(value, arguments, forceSingle)
```

boolean tryFillOptionShort(Enum value, List<String> arguments, boolean forceSingle)

```
null
null
null

## 📄 NutsXmlFormat
```

public net.vpc.app.nuts.NutsXmlFormat

```
Xml Format Helper class
### 🌐 Instance Properties
#### 🖊️compact
if true compact xml generated. if false, sue more versatile/formatted output.
```

[read-write[NutsXmlFormat compact boolean isCompact() NutsXmlFormat setCompact(compact)

```
#### 🌐 session
update session
```

[write-only[NutsXmlFormat session NutsXmlFormat setSession(session)

```
#### 📝 value  
set value to format
```

[read-write] NutsXmlFormat value Object getValue() NutsXmlFormat setValue(value)

```
### ⚙ Instance Methods  
### ⚙ configure(skipUnsupported, args)  
configure the current command with the given arguments. This is an  
override of the ``` NutsConfigurable#configure(boolean, java.lang.String...)```
```

to help return a more specific return type;

```
NutsXmlFormat configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ fromXmlElement(xmlElement, clazz)

convert `xmlElement` to a valid instance of type `clazz`

```
T fromXmlElement(Element xmlElement, Class<T> clazz)
```

xmlElement to convert target class

⚙️ parse(bytes, clazz)

parse bytes as xml to the given class

```
T parse(byte[] bytes, Class<T> clazz)
```

bytes to parse target class

⚙️ parse(file, clazz)

Parse Xml Content as given class type.

```
T parse(File file, Class<T> clazz)
```

input content type to parse to

parse(inputStream, clazz)

parse inputStream as xml to the given class

```
T parse(InputStream inputStream, Class<T> clazz)
```

inputStream to parse target class

parse(path, clazz)

Parse Xml Content as given class type.

```
T parse(Path path, Class<T> clazz)
```

input content type to parse to

parse(reader, clazz)

Parse Xml Content as given class type.

```
T parse(Reader reader, Class<T> clazz)
```

input content type to parse to

parse(url, clazz)

parse url content as xml to the given class

```
T parse(URL url, Class<T> clazz)
```

url to parse target class

⚙ **toXmlDocument(value)**

convert `value` to an xml document.

Document **toXmlDocument(Object value)**

value to convert

⚙ **toXmlElement(value, xmlDocument)**

convert `value` to a valid root element to add to the given `xmlDocument` .if the document is null, a new one will be created.

Element **toXmlElement(Object value, Document xmlDocument)**

value to convert target document

⚙ **value(value)**

set value to format

NutsXmlFormat **value(Object value)**

value to format

Input Output

NutsIOCopyAction

```
public net.vpc.app.nuts.NutsIOCopyAction
```

I/O Action that help monitored copy of one or multiple resource types. Implementation should at least handle the following types as valid sources :

- InputStream
- string (as path or url)
- File (file or directory)
- Path (file or directory)
- URL

and the following types as valid targets :

- OutputStream
- string (as path or url)
- File (file or directory)
- Path (file or directory)

Instance Properties

byteArrayResult

run this copy action with ` java.io.ByteArrayOutputStream` target and return bytes result

```
[read-only] byte[] byteArrayResult  
byte[] getByteArrayResult()
```

interruptible

mark created stream as interruptible so that one can call ` #interrupt()`

```
[read-write] NutsIOCopyAction interruptible  
boolean isInterruptible()  
NutsIOCopyAction setInterruptible(interruptible)
```

logProgress

switch log progress flag to `value`.

```
[read-write] NutsIOCopyAction logProgress  
boolean isLogProgress()  
NutsIOCopyAction setLogProgress(value)
```

progressMonitor

set progress monitor. Will create a singleton progress monitor factory

```
[write-only] NutsIOCopyAction progressMonitor  
NutsIOCopyAction setProgressMonitor(value)
```

progressMonitorFactory

set progress factory responsible of creating progress monitor

```
[read-write] NutsIOCopyAction progressMonitorFactory  
NutsProgressFactory getProgressMonitorFactory()  
NutsIOCopyAction setProgressMonitorFactory(value)
```

safe

switch safe copy flag to `value`

```
[read-write] NutsIOCopyAction safe  
boolean isSafe()  
NutsIOCopyAction setSafe(value)
```

 session

update current session

```
[read-write] NutsIOCopyAction session
NutsSession getSession()
NutsIOCopyAction setSession(session)
```

 skipRoot

set skip root flag to `value`

```
[read-write] NutsIOCopyAction skipRoot
boolean isSkipRoot()
NutsIOCopyAction setSkipRoot(value)
```

 source

update source to copy from

```
[read-write] NutsIOCopyAction source
Object getSource()
NutsIOCopyAction setSource(source)
```

 target

update target to copy from

```
[read-write] NutsIOCopyAction target
Object getTarget()
NutsIOCopyAction setTarget(target)
```

 validator

update validator

```
[read-write] NutsIOCopyAction validator  
NutsIOCopyValidator getValidator()  
NutsIOCopyAction setValidator.validator)
```

⚙ Instance Methods

⚙ from(source)

update source to copy from

```
NutsIOCopyAction from(Object source)
```

source to copy from

⚙ from(source)

update source to copy from

```
NutsIOCopyAction from(String source)
```

source to copy from

⚙ from(source)

update source to copy from

```
NutsIOCopyAction from(InputStream source)
```

source to copy from

⚙ from(source)

update source to copy from

```
NutsIOCopyAction from(File source)
```

source to copy from

⚙️ `from(source)`

update source to copy from

```
NutsIOCopyAction from(Path source)
```

source to copy from

⚙️ `from(source)`

update source to copy from

```
NutsIOCopyAction from(URL source)
```

source to copy from

⚙️ `interrupt()`

interrupt last created stream. An exception is throws when the stream is read.

```
NutsIOCopyAction interrupt()
```

⚙️ `logProgress()`

switch log progress flag to to true.

```
NutsIOCopyAction logProgress()
```

⚙️ `logProgress(value)`

switch log progress to ` value`

```
NutsIOCopyAction logProgress(boolean value)
```

log progress

⚙️ **progressMonitor(value)**

set progress monitor. Will create a singleton progress monitor factory

NutsIOCopyAction **progressMonitor(NutsProgressMonitor value)**

new value

⚙️ **progressMonitorFactory(value)**

set progress factory responsible of creating progress monitor

NutsIOCopyAction **progressMonitorFactory(NutsProgressFactory value)**

new value

⚙️ **run()**

run this copy action

NutsIOCopyAction **run()**

⚙️ **safe()**

arm safe copy flag

NutsIOCopyAction **safe()**

⚙️ **safe(value)**

switch safe copy flag to `value`

NutsIOCopyAction **safe(boolean value)**

value

⚙️ **skipRoot()**

set skip root flag to `true`

NutsIOCopyAction **skipRoot()**

⚙️ **skipRoot(value)**

set skip root flag to `value`

NutsIOCopyAction **skipRoot(boolean value)**

new value

⚙️ **to(target)**

update target

NutsIOCopyAction **to(Object target)**

target

⚙️ **to(target)**

update target

NutsIOCopyAction **to(OutputStream target)**

target

⚙️ **to(target)**

update target to copy from

```
NutsIOCopyAction to(String target)
```

target to copy to

⚙️ `to(target)`

update target to copy from

```
NutsIOCopyAction to(Path target)
```

target to copy to

⚙️ `to(target)`

update target to copy from

```
NutsIOCopyAction to(File target)
```

target to copy to

⚙️ `validator(validator)`

update validator

```
NutsIOCopyAction validator(NutsIOCopyValidator validator)
```

validator

📄 `NutsIODeleteAction`

```
public net.vpc.app.nuts.NutsIODeleteAction
```

I/O Action that help monitored delete.

Instance Properties

failFast

update fail fast flag

```
[read-write] NutsIODeleteAction failFast  
boolean isFailFast()  
NutsIODeleteAction setFailFast(failFast)
```

session

update session

```
[read-write] NutsIODeleteAction session  
NutsSession getSession()  
NutsIODeleteAction setSession(session)
```

target

update target to delete

```
[read-write] NutsIODeleteAction target  
Object getTarget()  
NutsIODeleteAction setTarget(target)
```

Instance Methods

failFast()

set fail fast flag

```
NutsIODeleteAction failFast()
```

failFast(failFast)

update fail fast flag

```
NutsIODeleteAction failFast(boolean failFast)
```

value

⚙️ [run\(\)](#)

run delete action and return `this`

```
NutsIODeleteAction run()
```

⚙️ [target\(target\)](#)

update target to delete

```
NutsIODeleteAction target(Object target)
```

target

☕️ [NutsIOWorkerAction](#)

```
public net.vpc.app.nuts.NutsIOWorkerAction
```

I/O command to work.

ⓘ [Instance Properties](#)

✍️ ⓘ [algorithm](#)

select hash algorithm.

```
[read-write] NutsIOWorkerAction algorithm
String getAlgorithm()
NutsIOWorkerAction setAlgorithm(algorithm)
```

⚙ Instance Methods

⚙ algorithm(algorithm)

select hash algorithm.

```
NutsIOHashAction algorithm(String algorithm)
```

hash algorithm. may be any algorithm supported by {@link MessageDigest#getInstance(String)} including 'MD5' and 'SHA1'

⚙ computeBytes()

compute hash digest and return it as byte array

```
byte[] computeBytes()
```

⚙ computeString()

compute hash digest and return it as hexadecimal string

```
String computeString()
```

⚙ md5()

select MD5 hash algorithm

```
NutsIOHashAction md5()
```

⚙ sha1()

select MD5 hash algorithm

```
NutsIOHashAction sha1()
```

 **source(descriptor)**

source stream to hash

```
NutsIOHashAction source(NutsDescriptor descriptor)
```

source descriptor to hash

 **source(file)**

file to hash

```
NutsIOHashAction source(File file)
```

source file to hash

 **source(input)**

source stream to hash

```
NutsIOHashAction source(InputStream input)
```

source stream to hash

 **source(path)**

file to hash

```
NutsIOHashAction source(Path path)
```

source path to hash

 **writeHash(out)**

compute hash and writes it to the output stream

```
NutsIOHashAction writeHash(OutputStream out)
```

output stream

✍ NutsIOLockAction

```
public net.vpc.app.nuts.NutsIOLockAction
```

Lock builder to create mainly File based Locks

Instance Properties

📝 resource

update resource

```
[read-write] NutsIOLockAction resource  
Object getResource()  
NutsIOLockAction setResource(source)
```

📝 session

update session

```
[read-write] NutsIOLockAction session  
NutsSession getSession()  
NutsIOLockAction setSession(session)
```

📝 source

update source

```
[read-write] NutsIOLockAction source  
Object getSource()  
NutsIOLockAction setSource(source)
```

⚙ Instance Methods

⚙ call(runnable)

create lock object for the given source and resource

```
T call(Callable<T> runnable)
```

runnable

⚙ call(runnable, time, unit)

create lock object for the given source and resource

```
T call(Callable<T> runnable, long time, TimeUnit unit)
```

runnable time unit

⚙ create()

create lock object for the given source and resource

```
NutsLock create()
```

⚙ resource(source)

update resource

```
NutsIOLockAction resource(Object source)
```

resource

⚙ run(runnable)

create lock object for the given source and resource

```
void run(Runnable runnable)
```

runnable

 **run(runnable, time, unit)**

create lock object for the given source and resource

```
void run(Runnable runnable, long time, TimeUnit unit)
```

runnable time unit

 **source(source)**

update source

```
NutsIOLockAction source(Object source)
```

source

 **NutsIOManager**

```
public net.vpc.app.nuts.NutsIOManager
```

I/O Manager supports a set of operations to manipulate terminals and files in a handy manner that is monitorable and Workspace aware.

 **Instance Properties**

 **systemTerminal**

update workspace wide system terminal

```
[read-write] NutsIOManager systemTerminal  
NutsSystemTerminal getSystemTerminal()  
NutsIOManager setSystemTerminal(terminal)
```

terminal

update workspace wide terminal

```
[read-write] NutsIOManager terminal  
NutsSessionTerminal getTerminal()  
NutsIOManager setTerminal(terminal)
```

terminalFormat

return terminal format that handles metrics and format/escape methods.

```
[read-only] NutsTerminalFormat terminalFormat  
NutsTerminalFormat getTerminalFormat()
```

Instance Methods

compress()

create new ` NutsIOCompressAction` instance

```
NutsIOCompressAction compress()
```

copy()

create new ` NutsIOCopyAction` instance

```
NutsIOCopyAction copy()
```

⚙️ `createApplicationContext(args, appClass, storeId, startTimeMillis)`

create a new instance of ` NutsApplicationContext`

```
NutsApplicationContext createApplicationContext(String[] args, Class appClass,  
String storeId, long startTimeMillis)
```

application arguments application class application store id or null application start time

⚙️ `createPrintStream(out, mode)`

create print stream that supports the given ` mode` .If the given ` out` is a PrintStream that supports ` mode` , it should bereturned without modification.

```
PrintStream createPrintStream(OutputStream out, NutsTerminalMode mode)
```

stream to wrap mode to support

⚙️ `createTempFile(name)`

create temp file in the workspace's temp folder

```
Path createTempFile(String name)
```

file name

⚙️ `createTempFile(name, repository)`

create temp file in the repository's temp folder

```
Path createTempFile(String name, NutsRepository repository)
```

file name repository

⚙️ `createTempFolder(name)`

create temp folder in the workspace's temp folder

Path **createTempFolder**(String name)

folder name

⚙ **createTempFolder**(name, repository)

create temp folder in the repository's temp folder

Path **createTempFolder**(String name, NutsRepository repository)

folder name repository

⚙ **createTerminal**()

return new terminal bound to system terminal

NutsSessionTerminal **createTerminal**()

⚙ **createTerminal**(parent)

return new terminal bound to the given `parent`

NutsSessionTerminal **createTerminal**(NutsTerminalBase parent)

parent terminal or null

⚙ **delete**()

create new `NutsIToDeleteAction` instance

NutsIToDeleteAction **delete**()

⚙ **executorService**()

return non null executor service

```
ExecutorService executorService()
```

⚙️ **expandPath(path)**

expand path to Workspace Location

```
String expandPath(String path)
```

path to expand

⚙️ **expandPath(path, baseFolder)**

expand path to `baseFolder` .Expansion mechanism supports '~' prefix (linux like) and will expand path to `baseFolder` if it was resolved as a relative path.

```
String expandPath(String path, String baseFolder)
```

path to expand base folder to expand relative paths to

⚙️ **hash()**

create new `NutsIOHashAction` instance that helpshashing streams and files.

```
NutsIOHashAction hash()
```

⚙️ **loadFormattedString(reader, classLoader)**

load resource as a formatted string to be used mostly as a help string.

```
String loadFormattedString(Reader reader, ClassLoader classLoader)
```

resource reader class loader

⚙️ **loadFormattedString(resourcePath, classLoader, defaultValue)**

load resource as a formatted string to be used mostly as a help string.

```
String loadFormattedString(String resourcePath, ClassLoader classLoader, String defaultValue)
```

resource path class loader default value if the loading fails

⚙️ **lock()**

create new ` NutsIOLockAction` instance

```
NutsIOLockAction lock()
```

⚙️ **monitor()**

create new ` NutsMonitorAction` instance that helps monitoring streams.

```
NutsMonitorAction monitor()
```

⚙️ **nullInputStream()**

create a null input stream instance

```
InputStream nullInputStream()
```

⚙️ **nullPrintStream()**

create a null print stream instance

```
PrintStream nullPrintStream()
```

⚙️ **parseExecutionEntries(file)**

parse Execution Entries

```
NutsExecutionEntry[] parseExecutionEntries(File file)
```

jar file

⚙ parseExecutionEntries(file)

parse Execution Entries

```
NutsExecutionEntry[] parseExecutionEntries(Path file)
```

jar file

⚙ parseExecutionEntries(inputStream, type, sourceName)

parse Execution Entries

```
NutsExecutionEntry[] parseExecutionEntries(InputStream inputStream, String type,  
String sourceName)
```

stream stream type stream source name (optional)

⚙ ps()

create new ` NutsIOProcessAction` instance

```
NutsIOProcessAction ps()
```

⚙ systemTerminal()

return terminal format that handles metrics and format/escape methods.

```
NutsSystemTerminal systemTerminal()
```

⚙ terminal()

return workspace default terminal

```
NutsSessionTerminal terminal()
```

⚙️ `terminalFormat()`

return terminal format that handles metrics and format/escape methods

```
NutsTerminalFormat terminalFormat()
```

⚙️ `uncompress()`

create new `NutsIOWorker` instance

```
NutsIOWorker uncompress()
```

☕️ `NutsLock`

```
public net.vpc.app.nuts.NutsLock
```

`NutsLock` is simply an adapter to standard `Lock`. It adds no extra functionality but rather is provided as a base for future changes.

☕️ `NutsNonFormattedPrintStream`

```
public net.vpc.app.nuts.NutsNonFormattedPrintStream
```

Non formated Print Stream Anchor Interface

☕️ `NutsOutputStreamTransparentAdapter`

```
public net.vpc.app.nuts.NutsOutputStreamTransparentAdapter
```

Interface to enable marking system streams. When creating new processes nuts will dereference `NutsOutputStreamTransparentAdapter` to check if the `OutputStream` is a system io. In that case nuts will "inherit" output/error stream

⚙ Instance Methods

⚙ `baseOutputStream()`

de-referenced stream

```
OutputStream baseOutputStream()
```

☕ NutsSystemTerminal

```
public net.vpc.app.nuts.NutsSystemTerminal
```

⚙ Instance Methods

⚙ `isStandardErrorStream(out)`

```
boolean isStandardErrorStream(OutputStream out)
```

null

⚙ `isStandardInputStream(in)`

```
boolean isStandardInputStream(InputStream in)
```

null

⚙ `isStandardOutputStream(out)`

```
boolean isStandardOutputStream(OutputStream out)
```

null

☕ NutsTerminal

```
public net.vpc.app.nuts.NutsTerminal
```

A Terminal handles in put stream, an output stream and an error stream to communicate with user.

Instance Properties

  errMode

change terminal mode for err

```
[read-write] NutsTerminal errMode  
NutsTerminalMode getErrMode()  
NutsTerminal setErrMode(mode)
```

  mode

change terminal mode for both out and err

```
[write-only] NutsTerminal mode  
NutsTerminal setMode(mode)
```

  outMode

change terminal mode for out

```
[read-write] NutsTerminal outMode  
NutsTerminalMode getOutMode()  
NutsTerminal setOutMode(mode)
```

Instance Methods

 ask()

create a `NutsQuestion` to write a question to the terminal's output stream and read a typed value from the terminal's input stream.

```
NutsQuestion<T> ask()
```

⚙ `err()`

return terminal's error stream

```
PrintStream err()
```

⚙ `errMode(mode)`

change terminal mode for out

```
NutsTerminal errMode(NutsTerminalMode mode)
```

mode

⚙ `in()`

return terminal's input stream

```
InputStream in()
```

⚙ `mode(mode)`

change terminal mode for both out and err

```
NutsTerminal mode(NutsTerminalMode mode)
```

mode

⚙ `out()`

return terminal's output stream

```
PrintStream out()
```

⚙ **outMode(mode)**

change terminal mode for out

```
NutsTerminal outMode(NutsTerminalMode mode)
```

mode

⚙ **readLine(promptFormat, params)**

Reads a single line of text from the terminal's input stream.

```
String readLine(String promptFormat, Object params)
```

prompt message format (cstyle) prompt message parameters

⚙ **readPassword(promptFormat, params)**

Reads password as a single line of text from the terminal's input stream.

```
char[] readPassword(String promptFormat, Object params)
```

prompt message format (cstyle) prompt message parameters

Internal

« PrivateNutsArgumentsParser

```
final net.vpc.app.nuts.PrivateNutsArgumentsParser
```

Nuts Arguments parser. Creates a ` NutsWorkspaceOptions` instance from string array of valid nuts options

« Static Methods

« parseLevel(s)

```
Level parseLevel(String s)
```

null

« parseLogLevel(logConfig, cmdLine, enabled)

```
void parseLogLevel(NutsLogConfig logConfig, NutsCommandLine cmdLine, boolean  
enabled)
```

null null null

« parseNutsArguments(bootArguments)

Create a ` NutsWorkspaceOptions` instance from string array of valid nuts options

```
NutsWorkspaceOptionsBuilder parseNutsArguments(String[] bootArguments)
```

input arguments to parse

« parseNutsArguments(bootArguments, options)

Fill a ` NutsWorkspaceOptions` instance from string array of valid nuts options

```
void parseNutsArguments(String[] bootArguments, NutsWorkspaceOptionsBuilder options)
```

input arguments to parse options instance to fill

⌚⚙️ **parseNutsStoreLocationLayout(s)**

```
NutsOsFamily parseNutsStoreLocationLayout(String s)
```

null

⌚⚙️ **parseNutsStoreLocationStrategy(s)**

```
NutsStoreLocationStrategy parseNutsStoreLocationStrategy(String s)
```

null

⌚⚙️ **parseNutsTerminalMode(s)**

```
NutsTerminalMode parseNutsTerminalMode(String s)
```

null

⌚⚙️ **parseNutsWorkspaceOpenMode(s)**

```
NutsWorkspaceOpenMode parseNutsWorkspaceOpenMode(String s)
```

null

Constructors

PrivateNutsArgumentsParser()

private constructor

```
PrivateNutsArgumentsParser()
```

↳ `PrivateNutsBootConfigLoader`

```
final net.vpc.app.nuts.PrivateNutsBootConfigLoader
```

JSON Config Best Effort Loader

↳ `Static Methods`

↳ `getApiVersionOrdinalNumber(s)`

```
int getApiVersionOrdinalNumber(String s)
```

null

↳ `loadBootConfig(workspaceLocation, LOG)`

```
PrivateNutsWorkspaceInitInformation loadBootConfig(String workspaceLocation,  
PrivateNutsLog LOG)
```

null null

↳ `loadBootConfigJSON(json, LOG)`

```
PrivateNutsWorkspaceInitInformation loadBootConfigJSON(String json,  
PrivateNutsLog LOG)
```

null null

↳ `loadConfigVersion502(config, jsonObject, LOG)`

best effort to load config object from jsonObject saved with nuts version "[0.5.2,0.5.6[".[]

```
void loadConfigVersion502(PrivateNutsWorkspaceInitInformation config, Map<String, Object> jsonObject, PrivateNutsLog LOG)
```

config object to fill config JSON object null

⌚⚙️ `loadConfigVersion506(config, jsonObject, LOG)`

best effort to load config object from jsonObject saved with nuts version "[0.5.6[" and later.

```
void loadConfigVersion506(PrivateNutsWorkspaceInitInformation config, Map<String, Object> jsonObject, PrivateNutsLog LOG)
```

config object to fill config JSON object null

⌚⚙️ `loadConfigVersion507(config, jsonObject, LOG)`

best effort to load config object from jsonObject saved with nuts version "0.5.6" and later.

```
void loadConfigVersion507(PrivateNutsWorkspaceInitInformation config, Map<String, Object> jsonObject, PrivateNutsLog LOG)
```

config object to fill config JSON object null

⌚️ `PrivateNutsBootWorkspaceFactoryComparator`

```
final net.vpc.app.nuts.PrivateNutsBootWorkspaceFactoryComparator
```

⌚ Instance Fields

⌚ `options`

```
private final NutsWorkspaceOptions options
```

Constructors

PrivateNutsBootWorkspaceFactoryComparator(options)

```
PrivateNutsBootWorkspaceFactoryComparator(NutsWorkspaceOptions options)
```

null

Instance Methods

compare(o1, o2)

```
int compare(NutsBootWorkspaceFactory o1, NutsBootWorkspaceFactory o2)
```

null null

PrivateNutsCommandLine

```
final net.vpc.app.nuts.PrivateNutsCommandLine
```

Simple Command line parser implementation. The command line supports arguments in the following forms :

- non option arguments : any argument that does not start with '-'

* long option arguments : any argument that starts with a single '--' in the form of

```
--[//][!]?[^=]*[=.*]
```

- // means disabling the option
- ! means switching (to 'false') the option's value
- the string before the '=' is the option's key
- the string after the '=' is the option's value

Examples :

- --!enable : option 'enable' with 'false' value
- --enable=yes : option 'enable' with 'yes' value
- --!enable=yes : invalid option (no error will be thrown but the result is undefined)

* simple option arguments : any argument that starts with a single '-' in the form of

```
-[//][!]?[a-z][=.*]
```

if expandSimpleOptions=false. When activating expandSimpleOptions, multi characters key will be expanded as multiple separate simple options Examples :

- -!enable (with expandSimpleOptions=false) : option 'enable' with 'false' value
- --enable=yes : option 'enable' with 'yes' value
- --!enable=yes : invalid option (no error will be thrown but the result is undefined)
- long option arguments : any argument that starts with a '--'

option may start with '!' to switch armed flags expandSimpleOptions : when activated

Constant Fields

NOT_SUPPORTED

```
private static final String NOT_SUPPORTED = "This a minimal implementation of  
NutsCommand used to bootstrap. This Method is not supported."
```

Static Methods

createArgument0(argument, eq)

```
NutsArgument createArgument0(String argument, char eq)
```

null null

⌚⚙️ `escapeArgument(arg)`

```
String escapeArgument(String arg)
```

null

⌚⚙️ `escapeArguments(args)`

```
String escapeArguments(String[] args)
```

null

⌚⚙️ `parseCommandLineArray(commandLineString)`

```
String[] parseCommandLineArray(String commandLineString)
```

null

⌚⚙️ `readEscapedArgument(charArray, i, sb)`

```
int readEscapedArgument(char[] charArray, int i, StringBuilder sb)
```

null null null

⌚ Instance Fields

⌚ `args`

```
private final LinkedList<String> args = new LinkedList<>()
```

⌚ `eq`

```
private final char eq = '='
```

[lookahead](#)

```
private final List<NutsArgument> lookahead = new ArrayList<>()
```

[Constructors](#)

[PrivateNutsCommandLine\(\)](#)

```
PrivateNutsCommandLine()
```

[PrivateNutsCommandLine\(args\)](#)

```
PrivateNutsCommandLine(String[] args)
```

null

[Instance Properties](#)

[arguments](#)

```
[write-only] NutsCommandLine public arguments  
public NutsCommandLine setArguments(arguments)
```

[autoComplete](#)

```
[read-write] NutsCommandLine public autoComplete  
private boolean isAutoComplete()  
public NutsCommandLine setAutoComplete(autoComplete)
```

[autoCompleteMode](#)

```
[read-only] public boolean autoCompleteMode  
public boolean isAutoCompleteMode()
```

 commandName

```
[read-write] String public commandName  
private String commandName  
public String getCommandName()  
public NutsCommandLine setCommandName(commandName)
```

 empty

```
[read-only] public boolean empty  
public boolean isEmpty()
```

 execMode

```
[read-only] public boolean execMode  
public boolean isExecMode()
```

 expandSimpleOptions

```
[read-write] boolean public expandSimpleOptions  
private boolean expandSimpleOptions = false  
public boolean isExpandSimpleOptions()  
public NutsCommandLine setExpandSimpleOptions(expand)
```

 specialSimpleOptions

```
[read-only] public Set<String> specialSimpleOptions  
private final Set<String> specialSimpleOptions = new HashSet<>()  
public String[] getSpecialSimpleOptions()
```

 wordIndex

```
[read-only] public int wordIndex  
private int wordIndex = 0  
public int getWordIndex()
```

⚙ Instance Methods

⚙ accept(values)

```
boolean accept(String values)
```

null

⚙ accept(index, values)

```
boolean accept(int index, String values)
```

null null

⚙ contains(name)

```
boolean contains(String name)
```

null

⚙ createArgument(argument)

```
NutsArgument createArgument(String argument)
```

null

⚙ createExpandedSimpleOption(start, negate, val)

```
String createExpandedSimpleOption(char start, boolean negate, char val)
```

null null null

⚙ createExpandedSimpleOption(start, negate, val)

```
String createExpandedSimpleOption(char start, boolean negate, String val)
```

null null null

⚙ **ensureNext(expandSimpleOptions, ignoreExistingExpanded)**

```
boolean ensureNext(boolean expandSimpleOptions, boolean ignoreExistingExpanded)
```

null null

⚙ **find(name)**

```
NutsArgument find(String name)
```

null

⚙ **get(index)**

```
NutsArgument get(int index)
```

null

⚙ **hasNext()**

```
boolean hasNext()
```

⚙ **highlightText(text)**

```
String highlightText(String text)
```

null

⚙️ **indexOf(name)**

```
int indexOf(String name)
```

null

⚙️ **isExpandableOption(v, expandSimpleOptions)**

```
boolean isExpandableOption(String v, boolean expandSimpleOptions)
```

null null

⚙️ **isNonOption(index)**

```
boolean isNonOption(int index)
```

null

⚙️ **isOption(index)**

```
boolean isOption(int index)
```

null

⚙️ **isPrefixed(nameSeqArray)**

```
boolean isPrefixed(String[] nameSeqArray)
```

null

⚙️ **isSpecialSimpleOption(option)**

```
boolean isSpecialSimpleOption(String option)
```

null

⚙ `length()`

```
int length()
```

⚙ `next()`

```
NutsArgument next()
```

⚙ `next(name)`

```
NutsArgument next(NutsArgumentName name)
```

null

⚙ `next(names)`

```
NutsArgument next(String names)
```

null

⚙ `next(expectValue, names)`

```
NutsArgument next(NutsArgumentType expectValue, String names)
```

null null

⚙ `next(required, expandSimpleOptions)`

```
NutsArgument next(boolean required, boolean expandSimpleOptions)
```

null null

⚙️ `next(name, forceNonOption, error)`

```
NutsArgument next(NutsArgumentName name, boolean forceNonOption, boolean error)
```

null null null

⚙️ `nextBoolean(names)`

```
NutsArgument nextBoolean(String names)
```

null

⚙️ `nextNonOption()`

```
NutsArgument nextNonOption()
```

⚙️ `nextNonOption(name)`

```
NutsArgument nextNonOption(NutsArgumentName name)
```

null

⚙️ `nextRequiredNonOption(name)`

```
NutsArgument nextRequiredNonOption(NutsArgumentName name)
```

null

⚙️ `nextString(names)`

```
NutsArgument nextString(String names)
```

null

⚙️ `parseLine(commandLine)`

```
NutsCommandLine parseLine(String commandLine)
```

null

⚙️ `peek()`

```
NutsArgument peek()
```

⚙️ `process(defaultConfigurable, processor)`

```
void process(NutsConfigurable defaultConfigurable, NutsCommandLineProcessor  
processor)
```

null null

⚙️ `pushBack(arg)`

```
NutsCommandLine pushBack(NutsArgument arg)
```

null

⚙️ `registerSpecialSimpleOption(option)`

```
NutsCommandLine registerSpecialSimpleOption(String option)
```

null

⚙️ `requireNonOption()`

```
NutsCommandLine requireNonOption()
```

⚙ required()

```
NutsCommandLine required()
```

⚙ required(errorMessage)

```
NutsCommandLine required(String errorMessage)
```

null

⚙ skip()

```
int skip()
```

⚙ skip(count)

```
int skip(int count)
```

null

⚙ skipAll()

```
int skipAll()
```

⚙ throwError(message)

```
void throwError(String message)
```

null

⚙ toArray()

```
String[] toArray()
```

⚙️ **toString()**

```
String toString()
```

⚙️ **unexpectedArgument()**

```
NutsCommandLine unexpectedArgument()
```

⚙️ **unexpectedArgument(errorMessage)**

```
NutsCommandLine unexpectedArgument(String errorMessage)
```

null

⚙️ **unregisterSpecialSimpleOption(option)**

```
NutsCommandLine unregisterSpecialSimpleOption(String option)
```

null

☕️ **PrivateNutsId**

```
final net.vpc.app.nuts.PrivateNutsId
```

simple and dummy implementation of NutsId base functions

☕️ ⚙️ **Static Methods**

☕️ ⚙️ **parse(id)**

```
PrivateNutsId parse(String id)
```

null

Constructors

PrivateNutsId(groupId, artifactId, version)

```
PrivateNutsId(String groupId, String artifactId, String version)
```

null null null

Instance Properties

artifactId

```
[read-only] public String artifactId  
private final String artifactId  
public String getArtifactId()
```

groupId

```
[read-only] public String groupId  
private final String groupId  
public String getGroupId()
```

longName

```
[read-only] public String longName  
public String getLongName()
```

shortName

```
[read-only] public String shortName  
public String getShortName()
```

version

```
[read-only] public String version  
private final String version  
public String getVersion()
```

⚙ Instance Methods

⚙ equals(obj)

```
boolean equals(Object obj)
```

null

⚙ hashCode()

```
int hashCode()
```

⚙ toString()

```
String toString()
```

☕ PrivateNutsJsonParser

```
final net.vpc.app.nuts.PrivateNutsJsonParser
```

(Very Simple) JSON parser

⌚⚙ Static Methods

⌚⚙ parse(path)

```
Map<String, Object> parse(Path path)
```

null

Instance Fields

st

```
private StreamTokenizer st
```

Constructors

PrivateNutsJsonParser(r)

```
PrivateNutsJsonParser(Reader r)
```

null

Instance Methods

nextArray()

```
List<Object> nextArray()
```

nextElement()

```
Object nextElement()
```

nextKeyValue()

```
Object[] nextKeyValue()
```

nextObject()

```
Map<String, Object> nextObject()
```

⚙ parse()

```
Object parse()
```

⚙ parseArray()

```
List<Object> parseArray()
```

⚙ parseObject()

```
Map<String, Object> parseObject()
```

⚙ readChar(expected)

```
void readChar(char expected)
```

null

⚙ str(a)

```
String str(int a)
```

null

☕ PrivateNutsLog

```
public net.vpc.app.nuts.PrivateNutsLog
```

⌚ Constant Fields

⌚ CACHE

```
public static final String CACHE = "CACHE"
```

⌚⌚ DEFAULT_DATE_TIME_FORMATTER

```
public static final DateTimeFormatter DEFAULT_DATE_TIME_FORMATTER =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS").withZone
(ZoneId.systemDefault())
```

⌚⌚ FAIL

```
public static final String FAIL = "FAIL"
```

⌚⌚ LOG_PARAM_PATTERN

```
private static final Pattern LOG_PARAM_PATTERN = Pattern.compile("\\{(?<v>[0-
9]+)\\}")
```

⌚⌚ READ

```
public static final String READ = "READ"
```

⌚⌚ START

```
public static final String START = "START"
```

⌚⌚ SUCCESS

```
public static final String SUCCESS = "SUCCESS"
```

⌚⌚ WARNING

```
public static final String WARNING = "WARNING"
```

Instance Properties

options

```
[write-only] NutsWorkspaceOptions public options  
private NutsWorkspaceOptions options  
public void setOptions(options)
```

Instance Methods

doLog(Level lvl, String logVerb, String s)

```
void doLog(Level lvl, String logVerb, String s)
```

null null null

isLoggable(Level lvl)

```
boolean isLoggable(Level lvl)
```

null

log(Level lvl, String logVerb, String message)

```
void log(Level lvl, String logVerb, String message)
```

null null null

log(Level lvl, String message, Throwable err)

```
void log(Level lvl, String message, Throwable err)
```

null null null

⚙️ `loglvl, logVerb, message, object)`

```
void log(Level lvl, String logVerb, String message, Object object)
```

null null null null

⚙️ `loglvl, logVerb, message, objects)`

```
void log(Level lvl, String logVerb, String message, Object[] objects)
```

null null null null

⌚ `PrivateNutsPlatformUtils`

```
final net.vpc.app.nuts.PrivateNutsPlatformUtils
```

⌚ Static Properties

⌚ `platformOsFamily`

```
[read-only] public static NutsOsFamily platformOsFamily  
public static NutsOsFamily getPlatformOsFamily()
```

⌚ ⚙️ Static Methods

⌚ ⚙️ `getPlatformGlobalHomeFolder(location, workspaceName)`

```
String getPlatformGlobalHomeFolder(NutsStoreLocation location, String  
workspaceName)
```

null null

⌚ ⚙️ `getPlatformHomeFolder(platformOsFamily, location, homeLocations, global, workspaceName)`

resolves nuts home folder.Home folder is the root for nuts folders.It depends on folder type and

store layout. For instance log folder depends on on the underlying operating system (linux,windows,...). Specifications: XDG Base Directory Specification (<https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html>)

```
String getPlatformHomeFolder(NutsOsFamily platformOsFamily, NutsStoreLocation location, Map<String, String> homeLocations, boolean global, String workspaceName)
```

location layout to resolve home for folder type to resolve home for workspace home locations global workspace workspace name or id (discriminator)

PrivateNutsTokenFilter

```
net.vpc.app.nuts.PrivateNutsTokenFilter
```

Instance Fields

expression

```
protected String expression
```

Constructors

PrivateNutsTokenFilter(expression)

```
PrivateNutsTokenFilter(String expression)
```

null

Instance Properties

blank

```
[read-only] public boolean blank  
public boolean isBlank()
```

📄 null

```
[read-only] public boolean null  
public booleanisNull()
```

⚙️ Instance Methods

⚙️ contains(substring)

```
boolean contains(String substring)
```

null

⚙️ like(pattern)

```
boolean like(String pattern)
```

null

⚙️ matches(pattern)

```
boolean matches(String pattern)
```

null

📥 PrivateNutsUtils

```
final net.vpc.app.nuts.PrivateNutsUtils
```

Created by vpc on 1/15/17.

💬 Constant Fields

DOLLAR_PLACEHOLDER_PATTERN

```
private static final Pattern DOLLAR_PLACEHOLDER_PATTERN = Pattern.compile
("[\$][{}](?<name>([a-zA-Z]+)){}]" )
```

NO_M2

```
public static final boolean NO_M2 = PrivateNutsUtils.getSysBoolNutsProperty
("no-m2", false)
```

Static Methods

compareRuntimeVersion(v1, v2)

v1 and v2 are supposed in the following form nbr1.nbr2, ... where all items (nbr) between dots are positive numbers

```
int compareRuntimeVersion(String v1, String v2)
```

version 1 version 2

copy(ff, to, LOG)

```
void copy(File ff, File to, PrivateNutsLog LOG)
```

null null null

copy(url, to, LOG)

```
void copy(URL url, File to, PrivateNutsLog LOG)
```

null null null

⌚⚙️ **copy**(from, to, closeInput, closeOutput)

```
long copy(InputStream from, OutputStream to, boolean closeInput, boolean  
closeOutput)
```

null null null null

⌚⚙️ **deleteAndConfirm**(directory, force, refForceAll, term, session, LOG)

```
boolean deleteAndConfirm(File directory, boolean force, ConfirmDelete  
refForceAll, NutsTerminal term, NutsSession session, PrivateNutsLog LOG)
```

null null null null null null

⌚⚙️ **deleteAndConfirmAll**(folders, force, header, term, session, LOG)

```
int deleteAndConfirmAll(File[] folders, boolean force, String header,  
NutsTerminal term, NutsSession session, PrivateNutsLog LOG)
```

null null null null null null

⌚⚙️ **deleteAndConfirmAll**(folders, force, refForceAll, header, term, session, LOG)

```
int deleteAndConfirmAll(File[] folders, boolean force, ConfirmDelete refForceAll,  
String header, NutsTerminal term, NutsSession session, PrivateNutsLog LOG)
```

null null null null null null null

⌚⚙️ **desc**(s)

```
String desc(Object s)
```

null

⌚⚙️ `formatLogValue(unresolved, resolved)`

```
String formatLogValue(Object unresolved, Object resolved)
```

null null

⌚⚙️ `formatPeriodMilli(period)`

```
String formatPeriodMilli(long period)
```

null

⌚⚙️ `formatRight(str, size)`

```
String formatRight(String str, int size)
```

null null

⌚⚙️ `formatURL(url)`

```
String formatURL(URL url)
```

null

⌚⚙️ `getAbsolutePath(path)`

```
String getAbsolutePath(String path)
```

null

⌚⚙️ `getSysBoolNutsProperty(property, defaultValue)`

```
boolean getSysBoolNutsProperty(String property, boolean defaultValue)
```

null null

⌚⚙️ `getSystemBoolean(property, defaultValue)`

```
Boolean getSystemBoolean(String property, Boolean defaultValue)
```

null null

⌚⚙️ `getSystemString(property, defaultValue)`

```
String getSystemString(String property, String defaultValue)
```

null null

⌚⚙️ `idToPath(id)`

```
String idToPath(PrivateNutsId id)
```

null

⌚⚙️ `isAbsolutePath(location)`

```
boolean isAbsolutePath(String location)
```

null

⌚⚙️ `isActualJavaCommand(cmd)`

```
boolean isActualJavaCommand(String cmd)
```

null

⌚⚙️ `isActualJavaOptions(options)`

```
boolean isActualJavaOptions(String options)
```

null

⌚⚙️ **isBlank**(str)

```
boolean isBlank(String str)
```

null

⌚⚙️ **isPreferConsole**(args)

```
boolean isPreferConsole(String[] args)
```

null

⌚⚙️ **isURL**(url)

```
boolean isURL(String url)
```

null

⌚⚙️ **isValidWorkspaceName**(workspace)

```
boolean isValidWorkspaceName(String workspace)
```

null

⌚⚙️ **loadURLProperties**(url, cacheFile, useCache, LOG)

```
Properties loadURLProperties(URL url, File cacheFile, boolean useCache,  
PrivateNutsLog LOG)
```

null null null null

⌚⚙️ **nvl(all)**

```
String nvl(Object all)
```

null

⌚⚙️ **parseBoolean(value, defaultValue)**

```
Boolean parseBoolean(String value, Boolean defaultValue)
```

null null

⌚⚙️ **parseDependencies(s)**

```
Set<String> parseDependencies(String s)
```

null

⌚⚙️ **parseFileSize(s)**

```
int parseFileSize(String s)
```

null

⌚⚙️ **readStringFromFile(file)**

```
String readStringFromFile(File file)
```

null

⌚⚙️ **readStringFromURL(requestURL)**

```
String readStringFromURL(URL requestURL)
```

null

⌚⚙️ `replaceDollarString(path, m)`

```
String replaceDollarString(String path, Function<String, String> m)
```

null null

⌚⚙️ `resolveJavaCommand(javaHome)`

```
String resolveJavaCommand(String javaHome)
```

null

⌚⚙️ `resolveValidWorkspaceName(workspace)`

```
String resolveValidWorkspaceName(String workspace)
```

null

⌚⚙️ `split(str, separators)`

```
List<String> split(String str, String separators)
```

null null

⌚⚙️ `split(str, separators, trim)`

```
List<String> split(String str, String separators, boolean trim)
```

null null null

⌚⚙️ `splitUrlStrings(repositories)`

```
List<String> splitUrlStrings(String repositories)
```

null

⌚⚙️ syspath(s)

```
String syspath(String s)
```

null

⌚⚙️ toFile(url)

```
File toFile(String url)
```

null

⌚⚙️ toFile(url)

```
File toFile(URL url)
```

null

⌚⚙️ to MavenFileName(nutId, extension)

```
String toMavenFileName(String nutId, String extension)
```

null null

⌚⚙️ toMavenPath(nutId)

```
String toMavenPath(String nutId)
```

null

⌚⚙️ trim(str)

```
String trim(String str)
```

null

⌚⚙️ trimToNull(str)

```
String trimToNull(String str)
```

null

⌚ PrivateNutsWorkspaceInitInformation

```
final net.vpc.app.nuts.PrivateNutsWorkspaceInitInformation
```

⌚ Instance Fields

⌚ workspace

```
private String workspace
```

⌚ Instance Properties

📝⌚ apild

```
[read-only] public String apiId  
public String getApiId()
```

📝⌚ apiVersion

workspace api version

```
[read-write] String public apiVersion  
private String apiVersion  
public String getApiVersion()  
public PrivateNutsWorkspaceInitInformation setApiVersion(apiVersion)
```

📝 bootClassWorldURLs

```
[write-only] URL[] public bootClassWorldURLs  
private URL[] bootClassWorldURLs  
public PrivateNutsWorkspaceInitInformation  
setBootClassWorldURLs(bootClassWorldURLs)
```

📝 bootRepositories

bootRepositories list (; separated) where to look for runtime dependencies

```
[read-write] String public bootRepositories  
private String bootRepositories  
public String getBootRepositories()  
public PrivateNutsWorkspaceInitInformation setBootRepositories(repositories)
```

📝 bootWorkspaceFactory

```
[read-write] NutsBootWorkspaceFactory public bootWorkspaceFactory  
private NutsBootWorkspaceFactory bootWorkspaceFactory  
public NutsBootWorkspaceFactory getBootWorkspaceFactory()  
public PrivateNutsWorkspaceInitInformation  
setBootWorkspaceFactory(bootWorkspaceFactory)
```

📝 cacheBoot

```
[read-only] public String cacheBoot  
public String getCacheBoot()
```

 classWorldLoader

```
[read-only] public ClassLoader classWorldLoader  
public ClassLoader getClassWorldLoader()
```

 classWorldURLs

```
[read-only] public URL[] classWorldURLs  
public URL[] getClassWorldURLs()
```

 extensionDependencies

```
[read-only] public String extensionDependencies  
public String getExtensionDependencies()
```

 extensionDependenciesSet

```
[read-write] LinkedHashSet<String> public extensionDependenciesSet  
private LinkedHashSet<String> extensionDependenciesSet  
public Set<String> getExtensionDependenciesSet()  
public PrivateNutsWorkspaceInitInformation  
setExtensionDependenciesSet(extensionDependenciesSet)
```

 extensionsSet

```
[read-write] LinkedHashSet<String> public extensionsSet  
private LinkedHashSet<String> extensionsSet  
public Set<String> getExtensionsSet()  
public void setExtensionsSet(extensionsSet)
```

 global

when global is true consider system wide folders (user independent but needs greater privileges)

```
[read-only] public boolean global  
private boolean global  
public boolean isGlobal()
```

homeLocations

workspace expected locations for all layout. Relevant when moving the workspace cross operating systems

```
[read-write] Map<String, String> public homeLocations  
private Map<String, String> homeLocations  
public Map<String, String> getHomeLocations()  
public void setHomeLocations(homeLocations)
```

javaCommand

java executable command to run nuts binaries

```
[read-write] String public javaCommand  
private String javaCommand  
public String getJavaCommand()  
public PrivateNutsWorkspaceInitInformation setJavaCommand(javaCommand)
```

javaOptions

java executable command options to run nuts binaries

```
[read-write] String public javaOptions  
private String javaOptions  
public String getJavaOptions()  
public PrivateNutsWorkspaceInitInformation setJavaOptions(javaOptions)
```

lib

```
[read-only] public String lib  
public String getLib()
```

 name

workspace name

```
[read-write] String public name
private String name
public String getName()
public void setName(name)
```

 options

```
[read-write] NutsWorkspaceOptions public options
private NutsWorkspaceOptions options
public NutsWorkspaceOptions getOptions()
public PrivateNutsWorkspaceInitInformation setOptions(options)
```

 repositoryStoreLocationStrategy

workspace bootRepositories store location strategy

```
[read-write] NutsStoreLocationStrategy public repositoryStoreLocationStrategy
private NutsStoreLocationStrategy repositoryStoreLocationStrategy
public NutsStoreLocationStrategy getRepositoryStoreLocationStrategy()
public PrivateNutsWorkspaceInitInformation
setRepositoryStoreLocationStrategy(repositoryStoreLocationStrategy)
```

 runtimeDependencies

```
[read-only] public String runtimeDependencies
public String getRuntimeDependencies()
```

 runtimeDependenciesSet

runtime artifact dependencies id list (; separated)

```
[read-write] LinkedHashSet<String> public runtimeDependenciesSet  
private LinkedHashSet<String> runtimeDependenciesSet  
public Set<String> getRuntimeDependenciesSet()  
public PrivateNutsWorkspaceInitInformation  
setRuntimeDependenciesSet(runtimeDependenciesSet)
```

runtimeld

workspace runtime id (group, name avec version)

```
[read-write] String public runtimeId  
private String runtimeId  
public String getRuntimeId()  
public PrivateNutsWorkspaceInitInformation setRuntimeId(runtimeId)
```

storeLocationLayout

workspace store location layout

```
[read-write] NutsOsFamily public storeLocationLayout  
private NutsOsFamily storeLocationLayout  
public NutsOsFamily getStoreLocationLayout()  
public PrivateNutsWorkspaceInitInformation  
setStoreLocationLayout(storeLocationLayout)
```

storeLocationStrategy

workspace store location strategy

```
[read-write] NutsStoreLocationStrategy public storeLocationStrategy  
private NutsStoreLocationStrategy storeLocationStrategy  
public NutsStoreLocationStrategy getStoreLocationStrategy()  
public PrivateNutsWorkspaceInitInformation  
setStoreLocationStrategy(storeLocationStrategy)
```

storeLocations

workspace store locations

```
[read-write] Map<String, String> public storeLocations
private Map<String, String> storeLocations
public Map<String, String> getStoreLocations()
public void setStoreLocations(storeLocations)
```



workspace uuid

```
[read-write] String public uuid
private String uuid
public String getUuid()
public void setUuid(uuid)
```



```
[write-only] ClassLoader public workspaceClassLoader
private ClassLoader workspaceClassLoader
public PrivateNutsWorkspaceInitInformation
setWorkspaceClassLoader(workspaceClassLoader)
```



```
[read-write] PrivateNutsWorkspaceInitInformation public workspaceLocation
public String getWorkspaceLocation()
public PrivateNutsWorkspaceInitInformation setWorkspaceLocation(workspace)
```

⚙ Instance Methods



```
String getStoreLocation(NutsStoreLocation location)
```

null

 **toString()**

```
String toString()
```

Logging

🖨️ NutsLogConfig

```
public net.vpc.app.nuts.NutsLogConfig
```

log configuration for running nuts

⌚ Constant Fields

⌚ serialVersionUID

```
private static final long serialVersionUID = 1
```

Constructors

⌚ NutsLogConfig()

```
NutsLogConfig()
```

⌚ NutsLogConfig(other)

```
NutsLogConfig(NutsLogConfig other)
```

null

Instance Properties

📝⌚ logFileBase

```
[read-write] String public logFileBase  
private String logFileBase = null  
public String getLogFileBase()  
public NutsLogConfig setLogFileBase(logFileBase)
```

 [logFileCount](#)

```
[read-write] int public logFileCount
private int logFileCount = 0
public int getLogFileCount()
public NutsLogConfig setLogFileCount(logFileCount)
```

 [logFileLevel](#)

```
[read-write] Level public logFileLevel
private Level logFileLevel = Level.OFF
public Level getLogFileLevel()
public NutsLogConfig setLogFileLevel(logFileLevel)
```

 [logFileName](#)

```
[read-write] String public logFileName
private String logFileName = null
public String getLogFileName()
public NutsLogConfig setLogFileName(logFileName)
```

 [logFileSize](#)

```
[read-write] int public logFileSize
private int logFileSize = 0
public int getLogFileSize()
public NutsLogConfig setLogFileSize(logFileSize)
```

 [logInherited](#)

```
[read-write] boolean public logInherited
private boolean logInherited = false
public boolean isLogInherited()
public NutsLogConfig setLogInherited(logInherited)
```

✍️ logTermLevel

```
[read-write] Level public logTermLevel  
private Level logTermLevel = Level.OFF  
public Level getLogTermLevel()  
public NutsLogConfig setLogTermLevel(logTermLevel)
```

⚙️ Instance Methods

⚙️ equals(o)

```
boolean equals(Object o)
```

null

⚙️ hashCode()

```
int hashCode()
```

⚙️ toString()

```
String toString()
```

☕️ NutsLogManager

```
public net.vpc.app.nuts.NutsLogManager
```

Nuts Log Manager

⌚ Instance Properties

✍️ fileHandler

file handler

```
[read-only] Handler fileHandler  
Handler getFileHandler()
```

fileLevel

return file logger level

```
[read-only] Level fileLevel  
Level getFileLevel()
```

handlers

Log handler

```
[read-only] Handler[] handlers  
Handler[] getHandlers()
```

termHandler

terminal handler

```
[read-only] Handler termHandler  
Handler getTermHandler()
```

termLevel

return terminal logger level

```
[read-only] Level termLevel  
Level getTermLevel()
```

Instance Methods

⚙️ addHandler(handler)

add the given handler

```
void addHandler(Handler handler)
```

handler to add

⚙️ of(clazz)

create an instance of ` NutsLogger`

```
NutsLogger of(Class clazz)
```

logger clazz

⚙️ of(name)

create an instance of ` NutsLogger`

```
NutsLogger of(String name)
```

logger name

⚙️ removeHandler(handler)

remove the given handler

```
void removeHandler(Handler handler)
```

handler to remove

⚙️ setFileLevel(level, options)

set file logger level

```
void setFileLevel(Level level, NutsUpdateOptions options)
```

new level update options

⚙ **setTermLevel**(level, options)

set terminal logger level

```
void setTermLevel(Level level, NutsUpdateOptions options)
```

new level update options

『 NutsLogger

```
public net.vpc.app.nuts.NutsLogger
```

Workspace aware Logger

⚙ Instance Methods

⚙ **isLoggable**(level)

Check if a message of the given level would actually be logged by this logger. This check is based on the Loggers effective level, which may be inherited from its parent.

```
boolean isLoggable(Level level)
```

a message logging level

⚙ **log**(record)

Log a LogRecord.

All the other logging methods in this class call through this method to actually perform any logging. Subclasses can override this single method to capture all log activity.

```
void log(LogRecord record)
```

the LogRecord to be published

```
⚙️ log(level, msg, thrown)
```

log message using 'FAIL' verb

```
void log(Level level, String msg, Throwable thrown)
```

message level message error thrown

```
⚙️ log(level, verb, msg)
```

log message using the given verb and level

```
void log(Level level, String verb, String msg)
```

message level message verb / category message

```
⚙️ log(level, verb, msgSupplier)
```

log message using the given verb and level

```
void log(Level level, String verb, Supplier<String> msgSupplier)
```

message level message verb / category message supplier

```
⚙️ log(level, verb, msg, params)
```

log message using the given verb and level

```
void log(Level level, String verb, String msg, Object params)
```

message level message verb / category message message parameters

⚙️ with()

create a logger op. A Logger Op handles all information to log in a custom manner.

NutsLoggerOp **with()**

☕ NutsLoggerOp

```
public net.vpc.app.nuts.NutsLoggerOp
```

Log operation

⚙️ Instance Methods

⚙️ error(error)

set log error

NutsLoggerOp **error(Throwable error)**

error thrown

⚙️ formatted()

set formatted mode (Nuts Stream Format)

NutsLoggerOp **formatted()**

⚙️ formatted(value)

set or unset formatted mode (Nuts Stream Format)

NutsLoggerOp **formatted(boolean value)**

formatted flag

⚙️ level(level)

set operation level

```
NutsLoggerOp level(Level level)
```

message level

⚙️ log(msgSupplier)

log the given message

```
void log(Supplier<String> msgSupplier)
```

message supplier

⚙️ log(msg, params)

log the given message

```
void log(String msg, Object params)
```

message message params

⚙️ style(style)

set message style (cstyle or positional)

```
NutsLoggerOp style(NutsTextFormatStyle style)
```

message format style

⚙️ time(time)

set operation time

```
NutsLoggerOp time(long time)
```

operation time in ms

⚙️ verb(verb)

set log verb

```
NutsLoggerOp verb(String verb)
```

verb or category

Other ArgEntry

```
net.vpc.app.nuts.ArgEntry
```

argument entry

 Instance Properties index

argument index

```
[read-only] int index
int getIndex()
```

 value

argument value

```
[read-only] String value
String getValue()
```

 ArgumentImpl

```
private final static net.vpc.app.nuts.ArgumentImpl
```

This is a minimal implementation of NutsArgument and hence should not be used. Instead an instance of NutsArgument can be retrieved using `NutsCommandLineFormat#createArgument(String)`

====

 Instance Fields

 eq

```
private final char eq
```

Constructors

ArgumentImpl(expression, eq)

Constructor

```
ArgumentImpl(String expression, char eq)
```

expression equals

Instance Properties

argumentKey

```
[read-only] public NutsArgument argumentKey  
public NutsArgument getArgumentKey()
```

argumentOptionName

```
[read-only] public NutsArgument argumentOptionName  
public NutsArgument getArgumentOptionName()
```

argumentValue

```
[read-only] public NutsArgument argumentValue  
public NutsArgument getArgumentValue()
```

blank

```
[read-only] public boolean blank  
public boolean isBlank()
```

boolean

```
[read-only] public boolean boolean  
public boolean isBoolean()
```

 booleanValue

```
[read-only] public boolean booleanValue  
public boolean getBooleanValue()
```

 double

```
[read-only] public double double  
public double getDouble()
```

 enabled

```
[read-only] public boolean enabled  
public boolean isEnabled()
```

 int

```
[read-only] public int int  
public int getInt()
```

 keyValue

```
[read-only] public boolean keyValue  
public boolean isKeyValue()
```

 keyValueSeparator

```
[read-only] public String keyValueSeparator  
public String getKeyValueSeparator()
```

 long

```
[read-only] public long long  
public long getLong()
```

 negated

```
[read-only] public boolean negated  
public boolean isNegated()
```

 nonOption

```
[read-only] public boolean nonOption  
public boolean isNonOption()
```

 null

```
[read-only] public boolean null  
public boolean isNull()
```

 option

true if expression starts with '-' or '+'

```
[read-only] public boolean option  
public boolean isOption()
```

 string

```
[read-only] public String string  
public String getString()
```

📄 stringKey

```
[read-only] public String stringKey  
public String getStringKey()
```

📄 stringOptionName

```
[read-only] public String stringOptionName  
public String getStringOptionName()
```

📄 stringOptionPrefix

```
[read-only] public String stringOptionPrefix  
public String getStringOptionPrefix()
```

📄 stringValue

```
[read-only] public String stringValue  
public String getStringValue()
```

📄 unsupported

true if the expression is a an option (starts with '-' or '+') but cannot not be evaluated.

```
[read-only] public boolean unsupported  
public boolean isUnsupported()
```

⚙ Instance Methods

⚙ getBoolean(defaultValue)

```
Boolean getBoolean(Boolean defaultValue)
```

null

⚙️ `getDouble(defaultValue)`

```
double getDouble(double defaultValue)
```

null

⚙️ `getInt(defaultValue)`

```
int getInt(int defaultValue)
```

null

⚙️ `getLong(defaultValue)`

```
long getLong(long defaultValue)
```

null

⚙️ `getString(defaultValue)`

```
String getString(String defaultValue)
```

null

⚙️ `getStringValue(defaultValue)`

```
String getStringValue(String defaultValue)
```

null

⚙️ `required()`

```
NutsArgument required()
```

⚙️ [toString\(\)](#)

```
String toString()
```

🖨️ [BootstrapURLs](#)

```
public static final net.vpc.app.nuts.BootstrapURLs
```

🔊 Constant Fields

🔊 LOCAL_MAVEN_CENTRAL

```
public static final String LOCAL_MAVEN_CENTRAL = "~/.m2/repository"
```

🔊 LOCAL_NUTS_FOLDER

```
public static final String LOCAL_NUTS_FOLDER = "/.vpc-public-nuts"
```

🔊 REMOTE_MAVEN_CENTRAL

```
public static final String REMOTE_MAVEN_CENTRAL =
"https://repo.maven.apache.org/maven2"
```

🔊 REMOTE_MAVEN_GIT

```
public static final String REMOTE_MAVEN_GIT =
"https://raw.githubusercontent.com/thevpc/vpc-public-maven/master"
```

🔊 REMOTE_NUTS_GIT

```
public static final String REMOTE_NUTS_GIT =
"https://raw.githubusercontent.com/thevpc/vpc-public-nuts/master"
```

Constructors

BootstrapURLs()

private constructor

```
BootstrapURLs()
```

ConfirmDelete

```
private net.vpc.app.nuts.ConfirmDelete
```

Instance Properties

force

```
[read-write] void force  
boolean isForce()  
void setForce(value)
```

Instance Methods

accept(directory)

```
boolean accept(File directory)
```

null

ignore(directory)

```
void ignore(File directory)
```

null

 Deps

```
public static net.vpc.app.nuts.Deps
```

 Instance Fields deps

```
LinkedHashSet<String> deps = new LinkedHashSet<>()
```

 repos

```
LinkedHashSet<String> repos = new LinkedHashSet<>()
```

 EnvEntry

```
net.vpc.app.nuts.EnvEntry
```

env entry

 Instance Properties  name

env name

```
[read-only] String name  
String getName()
```

  value

env value

```
[read-only] String value  
String getValue()
```

📄 Files

```
public static final net.vpc.app.nuts.Files
```

file related constants

CONSTANT FIELDS

Descriptor File Extension

```
public static final String DESCRIPTOR_FILE_EXTENSION = ".nuts"
```

Descriptor File Name

```
public static final String DESCRIPTOR_FILE_NAME = "nuts.json"
```

Nuts Command File Extension

```
public static final String NUTS_COMMAND_FILE_EXTENSION = ".nuts-cmd-  
alias.json"
```

Repository Configuration File Name

```
public static final String REPOSITORY_CONFIG_FILE_NAME = "nuts-  
repository.json"
```

Workspace API Configuration File Name

```
public static final String WORKSPACE_API_CONFIG_FILE_NAME = "nuts-api-  
config.json"
```

⌚ WORKSPACE_CONFIG_FILE_NAME

```
public static final String WORKSPACE_CONFIG_FILE_NAME = "nuts-workspace.json"
```

⌚ WORKSPACE_EXTENSION_CACHE_FILE_NAME

```
public static final String WORKSPACE_EXTENSION_CACHE_FILE_NAME = "nuts-extension-cache.json"
```

⌚ WORKSPACE_RUNTIME_CACHE_FILE_NAME

```
public static final String WORKSPACE_RUNTIME_CACHE_FILE_NAME = "nuts-runtime-cache.json"
```

Constructors

Files()

private constructor

```
Files()
```

Folders

```
public static final net.vpc.app.nuts.Folders
```

default folder names

Constant Fields

BOOT

```
public static final String BOOT = "boot"
```

⌚ ID

```
public static final String ID = "id"
```

⌚ REPOSITORIES

```
public static final String REPOSITORIES = "repos"
```

⌚ Constructors

⌚ Folders()

private constructor

```
Folders()
```

☕ IdProperties

```
public static final net.vpc.app.nuts.IdProperties
```

Nuts Id query parameter names. Nuts id has the following form
namespace://group:name#version?query where query is in the form key=value ` @ ` key=value...

This class defines all standard key names and their default values in the query part.

⌚ Constant Fields

⌚ ARCH

```
public static final String ARCH = "arch"
```

⌚ CLASSIFIER

```
public static final String CLASSIFIER = "classifier"
```

⌚ EXCLUSIONS

```
public static final String EXCLUSIONS = "exclusions"
```

⌚ FACE

```
public static final String FACE = "face"
```

⌚ NAMESPACE

```
public static final String NAMESPACE = "namespace"
```

⌚ OPTIONAL

```
public static final String OPTIONAL = "optional"
```

⌚ OS

```
public static final String OS = "os"
```

⌚ OSDIST

```
public static final String OSDIST = "osdist"
```

⌚ PACKAGING

```
public static final String PACKAGING = "packaging"
```

🔊 PLATFORM

```
public static final String PLATFORM = "platform"
```

🔊 SCOPE

```
public static final String SCOPE = "scope"
```

🔊 VERSION

```
public static final String VERSION = "version"
```

🔗 Constructors

🔗 IdProperties()

private constructor

```
IdProperties()
```

☕ Ids

```
public static final net.vpc.app.nuts.Ids
```

identifier related constants

🔊 Constant Fields

🔊 NUTS_API

```
public static final String NUTS_API = "net.vpc.app.nuts:nuts"
```

 NUTS_RUNTIME

```
public static final String NUTS_RUNTIME = "net.vpc.app.nuts:nuts-core"
```

 NUTS_SHELL

```
public static final String NUTS_SHELL = "net.vpc.app.nuts.toolbox:nsh"
```

 Constructors Ids()

private constructor

```
Ids()
```

 Mvn

```
public static net.vpc.app.nuts.Mvn
```

 Static Methods createFile(parent, child)

```
File createFile(String parent, String child)
```

null null

 isInfiniteLoopThread(className, methodName)

```
boolean isInfiniteLoopThread(String className, String methodName)
```

null null

⌚⚙️ `loadDependencies(rid, LOG, repos)`

Deps `LoadDependencies(PrivateNutsId rid, PrivateNutsLog LOG, Collection<String> repos)`

null null null

⌚⚙️ `loadDependencies(urlPath, LOG, repos)`

Deps `LoadDependencies(String urlPath, PrivateNutsLog LOG, Collection<String> repos)`

null null null

⌚⚙️ `loadDependenciesAndRepositoriesFromPomUrl(url, LOG)`

Deps `LoadDependenciesAndRepositoriesFromPomUrl(String url, PrivateNutsLog LOG)`

null null

⌚⚙️ `resolveLatestMavenId(zId, filter, LOG, bootRepositories)`

find latest maven artifact

String `ResolveLatestMavenId(PrivateNutsId zId, Predicate<String> filter, PrivateNutsLog LOG, Collection<String> bootRepositories)`

null filter null null

⌚⚙️ `resolveMavenFullPath(repo, nutsId, ext)`

String `ResolveMavenFullPath(String repo, String nutsId, String ext)`

null null null

 `resolveOrDownloadJar(nutsId, repositories, cacheFolder, LOG, includeDesc)`

```
File resolveOrDownloadJar(String nutsId, String[] repositories, String
cacheFolder, PrivateNutsLog LOG, boolean includeDesc)
```

null null null null null

 Names

```
public static final net.vpc.app.nuts.Names
```

name constants

 Constant Fields

 DEFAULT_REPOSITORY_NAME

```
public static final String DEFAULT_REPOSITORY_NAME = "local"
```

 DEFAULT_WORKSPACE_NAME

```
public static final String DEFAULT_WORKSPACE_NAME = "default-workspace"
```

 Constructors

 Names()

private constructor

```
Names()
```

 NutsApplicationLifeCycleImpl

```
private static net.vpc.app.nuts.NutsApplicationLifeCycleImpl
```

Default NutsApplicationLifeCycle implementation based on NutsApplication class.

Instance Fields

app

```
private final NutsApplication app
```

Constructors

NutsApplicationLifeCycleImpl(app)

application

```
NutsApplicationLifeCycleImpl(NutsApplication app)
```

application

Instance Methods

createApplicationContext(ws, args, startTimeMillis)

```
NutsApplicationContext createApplicationContext(NutsWorkspace ws, String[] args,  
long startTimeMillis)
```

null null null

onInstallApplication(applicationContext)

```
void onInstallApplication(NutsApplicationContext applicationContext)
```

null

onRunApplication(applicationContext)

```
void onRunApplication(NutsApplicationContext applicationContext)
```

null

⚙ **onUninstallApplication**(applicationContext)

```
void onUninstallApplication(NutsApplicationContext applicationContext)
```

null

⚙ **onUpdateApplication**(applicationContext)

```
void onUpdateApplication(NutsApplicationContext applicationContext)
```

null

⚙ **toString()**

```
String toString()
```

👉 **NutsArtifactCallBuilder**

```
public net.vpc.app.nuts.NutsArtifactCallBuilder
```

NutsArtifactCallBuilder is responsible of building instances of ` NutsArtifactCall` to be used as NutsDescriptor executor or installer. To get an instance of NutsArtifactCallBuilder you can use ` workspace.descriptor().callBuilder() ` ===== Instance Properties

📝 arguments

update arguments

```
[read-write] NutsArtifactCallBuilder arguments  
String[] getArguments()  
NutsArtifactCallBuilder setArguments(value)
```



update artifact id

```
[read-write] NutsArtifactCallBuilder id  
NutsId getId()  
NutsArtifactCallBuilder setId(value)
```



update call properties map (replace all existing properties)

```
[read-write] NutsArtifactCallBuilder properties  
Map<String, String> getProperties()  
NutsArtifactCallBuilder setProperties(value)
```

⚙ Instance Methods



create an immutable instance of `NutsArtifactCall` initialized with all of this attributes.

```
NutsArtifactCall build()
```



reset this instance to default (null) values

```
NutsArtifactCallBuilder clear()
```

⚙️ set(value)

initialize this instance from the given value

```
NutsArtifactCallBuilder set(NutsArtifactCallBuilder value)
```

copy from value

⚙️ set(value)

initialize this instance from the given value

```
NutsArtifactCallBuilder set(NutsArtifactCall value)
```

copy from value

🖨️ NutsCommandAutoComplete

```
public net.vpc.app.nuts.NutsCommandAutoComplete
```

Auto Complete Helper class used to collect argument candidates

⌚ Instance Properties

📄⌚ candidates

current candidates

```
[read-only] List<NutsArgumentCandidate> candidates  
List<NutsArgumentCandidate> getCandidates()
```

📄⌚ currentWordIndex

candidates index

```
[read-only] int currentWordIndex  
int getCurrentWordIndex()
```

line

command line string

```
[read-only] String line  
String getLine()
```

session

```
[read-only] NutsSession session  
NutsSession getSession()
```

words

command line arguments

```
[read-only] List<String> words  
List<String> getWords()
```

workspace

```
[read-only] NutsWorkspace workspace  
NutsWorkspace getWorkspace()
```

Instance Methods

addCandidate(value)

add candidate

```
void addCandidate(NutsArgumentCandidate value)
```

candidate

⚙️ `get(t)`

```
T get(Class<T> t)
```

null

⚠️ `NutsIOCopyValidationException`

```
public net.vpc.app.nuts.NutsIOCopyValidationException
```

Exception thrown when copy validation fails

🔗 Constructors

🔗 `NutsIOCopyValidationException(workspace)`

Constructs a new Validation Exception

```
NutsIOCopyValidationException(NutsWorkspace workspace)
```

null

🔗 `NutsIOCopyValidationException(workspace, cause)`

Constructs a new Validation Exception

```
NutsIOCopyValidationException(NutsWorkspace workspace, Throwable cause)
```

null cause

🔗 `NutsIOCopyValidationException(workspace, message)`

Constructs a new Validation Exception

```
NutsIOCopyValidationException(NutsWorkspace workspace, String message)
```

null message

⌚ NutsIOCopyValidationException(workspace, message, cause)

Constructs a new Validation Exception

```
NutsIOCopyValidationException(NutsWorkspace workspace, String message, Throwable cause)
```

null message cause

☕ NutsIOCopyValidator

```
public net.vpc.app.nuts.NutsIOCopyValidator
```

classes implementing this interface should check the validity of the stream that was copied.

⚙ Instance Methods

⚙ validate(targetContent)

Check the validity of the stream that was copied.

```
void validate(InputStream targetContent)
```

targetContent

☕ NutsIOProcessAction

```
public net.vpc.app.nuts.NutsIOProcessAction
```

I/O Action that help monitoring processes

Instance Properties

failFast

update fail fast flag

```
[read-write] NutsIOProcessAction failFast  
boolean isFailFast()  
NutsIOProcessAction setFailFast(failFast)
```

resultList

list all processes of type `#getType()`

```
[read-only] NutsResultList<NutsProcessInfo> resultList  
NutsResultList<NutsProcessInfo> getResultSet()
```

session

update session

```
[read-write] NutsIOProcessAction session  
NutsSession getSession()  
NutsIOProcessAction setSession(session)
```

type

set process type to consider. Supported 'java' or 'java#version'

```
[read-write] NutsIOProcessAction type  
String getType()  
NutsIOProcessAction setType(processType)
```

Instance Methods

⚙️ **failFast()**

set fail fast flag

```
NutsIOProcessAction failFast()
```

⚙️ **failFast(failFast)**

update fail fast flag

```
NutsIOProcessAction failFast(boolean failFast)
```

value

⚙️ **type(processType)**

set process type to consider. Supported 'java' or 'java#version'

```
NutsIOProcessAction type(String processType)
```

new type

☕️ **NutsIOUNcompressAction**

```
public net.vpc.app.nuts.NutsIOUNcompressAction
```

I/O Action that help monitored uncompress of one or multiple resource types.

ⓘ Instance Properties

📝 ⓘ **format**

update format

```
[read-write] NutsIOUncompressAction format
String getFormat()
NutsIOUncompressAction setFormat(format)
```

logProgress

switch log progress flag to `value`.

```
[read-write] NutsIOUncompressAction logProgress
boolean isLogProgress()
NutsIOUncompressAction setLogProgress(value)
```

progressMonitor

set progress monitor. Will create a singleton progress monitor factory

```
[write-only] NutsIOUncompressAction progressMonitor
NutsIOUncompressAction setProgressMonitor(value)
```

progressMonitorFactory

set progress factory responsible of creating progress monitor

```
[read-write] NutsIOUncompressAction progressMonitorFactory
NutsProgressFactory getProgressMonitorFactory()
NutsIOUncompressAction setProgressMonitorFactory(value)
```

safe

switch safe flag to `value`

```
[read-write] NutsIOUncompressAction safe
boolean isSafe()
NutsIOUncompressAction setSafe(value)
```

session

update current session

```
[read-write] NutsIOUncompressAction session  
NutsSession getSession()  
NutsIOUncompressAction setSession(session)
```

skipRoot

set skip root flag to `value`

```
[read-write] NutsIOUncompressAction skipRoot  
boolean isSkipRoot()  
NutsIOUncompressAction setSkipRoot(value)
```

source

update source to uncompress from

```
[read-write] NutsIOUncompressAction source  
Object getSource()  
NutsIOUncompressAction setSource(source)
```

target

update target

```
[read-write] NutsIOUncompressAction target  
Object getTarget()  
NutsIOUncompressAction setTarget(target)
```

Instance Methods

from(source)

update source to uncompress from

```
NutsIOUncompressAction from(InputStream source)
```

source to uncompress from

 **from**(source)

update source to uncompress from

```
NutsIOUncompressAction from(File source)
```

source to uncompress from

 **from**(source)

update source to uncompress from

```
NutsIOUncompressAction from(Path source)
```

source to uncompress from

 **from**(source)

update source to uncompress from

```
NutsIOUncompressAction from(URL source)
```

source to uncompress from

 **from**(source)

update source to uncompress from

```
NutsIOUncompressAction from(String source)
```

source to uncompress from

⚙️ **from(source)**

update source to uncompress from

```
NutsIOUncompressAction from(Object source)
```

source to uncompress from

⚙️ **getFormatOption(option)**

return format option

```
Object getFormatOption(String option)
```

option name

⚙️ **logProgress()**

switch log progress flag to true.

```
NutsIOUncompressAction logProgress()
```

⚙️ **logProgress(value)**

switch log progress flag to ` value` .

```
NutsIOUncompressAction logProgress(boolean value)
```

value

⚙️ **progressMonitor(value)**

set progress monitor. Will create a singleton progress monitor factory

```
NutsIOUncompressAction progressMonitor(NutsProgressMonitor value)
```

new value

⚙️ **progressMonitorFactory(value)**

set progress factory responsible of creating progress monitor

```
NutsIOUncompressAction progressMonitorFactory(NutsProgressFactory value)
```

new value

⚙️ **run()**

run this uncompress action

```
NutsIOUncompressAction run()
```

⚙️ **safe()**

arm safe flag

```
NutsIOUncompressAction safe()
```

⚙️ **safe(value)**

switch safe flag to `value`

```
NutsIOUncompressAction safe(boolean value)
```

value

⚙️ **setFormatOption(option, value)**

update format option

```
NutsIOUncompressAction setFormatOption(String option, Object value)
```

option name value

⚙️ **skipRoot()**

set skip root flag to `true`

NutsIOUncompressAction **skipRoot()**

⚙️ **skipRoot(value)**

set skip root flag to `value`

NutsIOUncompressAction **skipRoot(boolean value)**

new value

⚙️ **to(target)**

update target

NutsIOUncompressAction **to(String target)**

target

⚙️ **to(target)**

update target

NutsIOUncompressAction **to(Path target)**

target

⚙️ **to(target)**

update target

```
NutsIOUncompressAction to(File target)
```

target

 **to**(target)

update target

```
NutsIOUncompressAction to(Object target)
```

target

 NutsInfoFormat

```
public net.vpc.app.nuts.NutsInfoFormat
```

this class is responsible of displaying general information about the current workspace and repositories. It is invoked by the "info" standard command.

 Instance Properties

 **fancy**

enable fancy (custom, pretty) display mode

```
[read-write] NutsInfoFormat fancy
boolean isFancy()
NutsInfoFormat setFancy(fancy)
```

 session

update session

```
[write-only] NutsInfoFormat session
NutsInfoFormat setSession(session)
```

showRepositories

enable or disable display of all repositories information

```
[read-write] NutsInfoFormat showRepositories  
boolean isShowRepositories()  
NutsInfoFormat setShowRepositories(enable)
```

Instance Methods

addProperties(customProperties)

include custom properties from the given map

```
NutsInfoFormat addProperties(Map<String, String> customProperties)
```

custom properties

addProperty(key, value)

include a custom property

```
NutsInfoFormat addProperty(String key, String value)
```

custom property key custom property value

configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsInfoFormat configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ showRepositories()

enable display of all repositories information

```
NutsInfoFormat showRepositories()
```

⚙️ showRepositories(enable)

enable or disable display of all repositories information

```
NutsInfoFormat showRepositories(boolean enable)
```

if true enable

⌚ NutsLockBarrierException

```
public net.vpc.app.nuts.NutsLockBarrierException
```

Exception Thrown when a locked object is invoked.

Constructors

⌚ NutsLockBarrierException(workspace, lockedObject, lockObject)

Constructs a new lock exception.

```
NutsLockBarrierException(NutsWorkspace workspace, Object lockedObject, Object lockObject)
```

workspace locked Object lock Object

⌚ NutsLockBarrierException(workspace, message, lockedObject, lockObject)

Constructs a new lock exception.

```
NutsLockBarrierException(NutsWorkspace workspace, String message, Object  
lockedObject, Object lockObject)
```

workspace message or null locked Object lock Object

[🔗 NutsLockBarrierException\(workspace, message, lockedObject, lockObject, cause\)](#)

Constructs a new lock exception.

```
NutsLockBarrierException(NutsWorkspace workspace, String message, Object  
lockedObject, Object lockObject, Throwable cause)
```

workspace message or null locked Object lock Object cause

 [NutsLockReleaseException](#)

```
public net.vpc.app.nuts.NutsLockReleaseException
```

Exception Thrown when a locked object is invoked.

[🔗 Constructors](#)

[🔗 NutsLockReleaseException\(workspace, lockedObject, lockObject\)](#)

Constructs a new ock exception.

```
NutsLockReleaseException(NutsWorkspace workspace, Object lockedObject, Object  
lockObject)
```

workspace locked Object lock Object

[🔗 NutsLockReleaseException\(workspace, message, lockedObject, lockObject\)](#)

Constructs a new ock exception.

```
NutsLockReleaseException(NutsWorkspace workspace, String message, Object
lockedObject, Object lockObject)
```

workspace message or null locked Object lock Object

[🔗 NutsLockReleaseException\(workspace, message, lockedObject, lockObject, cause\)](#)

Constructs a new ock exception.

```
NutsLockReleaseException(NutsWorkspace workspace, String message, Object
lockedObject, Object lockObject, Throwable cause)
```

workspace message or null locked Object lock Object cause

 [NutsMonitorAction](#)

```
public net.vpc.app.nuts.NutsMonitorAction
```

Monitor action enables monitoring a long lasting operation such as copying a big file.

[🔗 Instance Properties](#)

 [length](#)

update operation length

```
[read-write] NutsMonitorAction length
long getLength()
NutsMonitorAction setLength(len)
```

 [logProgress](#)

when true, will include default factory (console) even if progressFactory is defined

```
[read-write] NutsMonitorAction logProgress  
boolean isLogProgress()  
NutsMonitorAction setLogProgress(value)
```



update action name

```
[read-write] NutsMonitorAction name  
String getName()  
NutsMonitorAction setName(name)
```



update action source origin

```
[read-write] NutsMonitorAction origin  
Object getOrigin()  
NutsMonitorAction setOrigin(origin)
```



set progress factory responsible of creating progress monitor

```
[read-write] NutsMonitorAction progressFactory  
NutsProgressFactory getProgressFactory()  
NutsMonitorAction setProgressFactory(value)
```



set progress monitor. Will create a singleton progress monitor factory

```
[write-only] NutsMonitorAction progressMonitor  
NutsMonitorAction setProgressMonitor(value)
```

session

update current session

```
[read-write] NutsMonitorAction session  
NutsSession getSession()  
NutsMonitorAction setSession(session)
```

source

update operation source

```
[write-only] NutsMonitorAction source  
NutsMonitorAction setSource(path)
```

Instance Methods

create()

Create monitored input stream

```
InputStream create()
```

length(len)

update operation length

```
NutsMonitorAction length(long len)
```

operation length

logProgress()

will include default factory (console) even if progressFactory is defined

```
NutsMonitorAction logProgress()
```

 **logProgress(value)**

when true, will include default factory (console) even if progressFactory is defined

```
NutsMonitorAction logProgress(boolean value)
```

value

 **name(name)**

update action name

```
NutsMonitorAction name(String name)
```

action name

 **origin(origin)**

update action source origin

```
NutsMonitorAction origin(Object origin)
```

source origin

 **progressFactory(value)**

set progress factory responsible of creating progress monitor

```
NutsMonitorAction progressFactory(NutsProgressFactory value)
```

new value

 **progressMonitor(value)**

set progress monitor. Will create a singleton progress monitor factory

```
NutsMonitorAction progressMonitor(NutsProgressMonitor value)
```

new value

⚙️ `source(path)`

update operation source

```
NutsMonitorAction source(String path)
```

operation source

⚙️ `source(path)`

update operation source

```
NutsMonitorAction source(Path path)
```

operation source

⚙️ `source(path)`

update operation source

```
NutsMonitorAction source(File path)
```

operation source

⚙️ `source(path)`

update operation source TODO: should this handle only streams?

```
NutsMonitorAction source(InputStream path)
```

operation source

☕ NutsProgressEvent

```
public net.vpc.app.nuts.NutsProgressEvent
```

Progress event

⌚ Instance Properties

⌚⌚ currentValue

progress current value

```
[read-only] long currentValue  
long getCurrentValue()
```

⌚⌚ error

error or null

```
[read-only] Throwable error  
Throwable getError()
```

⌚⌚ indeterminate

when true, max value is unknown, and the progress is indeterminate

```
[read-only] boolean indeterminate  
boolean isIndeterminate()
```

⌚⌚ maxValue

progress max value or -1 if intermediate

```
[read-only] long maxValue  
long getMaxValue()
```

📄 message

event message

```
[read-only] String message  
String getMessage()
```

📄 partialMillis

progress time from the starting of the last mark point.

```
[read-only] long partialMillis  
long getPartialMillis()
```

📄 partialValue

progress value from the last mark point. Mark point occurs when `NutsProgressMonitor#onProgress(NutsProgressEvent)` return false.

```
[read-only] long partialValue  
long getPartialValue()
```

📄 percent

progress percentage ([0..100])

```
[read-only] float percent  
float getPercent()
```

📄 session

Nuts Session

```
[read-only] NutsSession session  
NutsSession getSession()
```

source

progress source object

```
[read-only] Object source  
Object getSource()
```

timeMillis

progress time from the starting of the progress.

```
[read-only] long timeMillis  
long getTimeMillis()
```

NutsRepositoryRef

```
public net.vpc.app.nuts.NutsRepositoryRef
```

Constant Fields

serialVersionUID

```
private static final long serialVersionUID = 1
```

Constructors

NutsRepositoryRef()

```
NutsRepositoryRef()
```

NutsRepositoryRef(other)

```
NutsRepositoryRef(NutsRepositoryRef other)
```

null

 NutsRepositoryRef(name, location, deployPriority, enabled)

```
NutsRepositoryRef(String name, String location, int deployPriority, boolean  
enabled)
```

null null null null

 Instance Properties

  deployOrder

```
[read-write] int public deployOrder  
private int deployOrder  
public int getDeployOrder()  
public NutsRepositoryRef setDeployOrder(deployPriority)
```

  enabled

```
[read-write] boolean public enabled  
private boolean enabled = true  
public boolean isEnabled()  
public NutsRepositoryRef setEnabled(enabled)
```

  failSafe

```
[read-write] boolean public failSafe  
private boolean failSafe = false  
public boolean isFailSafe()  
public NutsRepositoryRef setFailSafe(failSafe)
```

  location

```
[read-write] String public location  
private String location  
public String getLocation()  
public NutsRepositoryRef setLocation(location)
```

✍️ name

```
[read-write] String public name  
private String name  
public String getName()  
public NutsRepositoryRef setName(name)
```

⚙️ Instance Methods

⚙️ copy()

```
NutsRepositoryRef copy()
```

⚙️ equals(obj)

```
boolean equals(Object obj)
```

null

⚙️ hashCode()

```
int hashCode()
```

⚙️ toString()

```
String toString()
```

☕ NutsRepositorySecurityManager

```
public net.vpc.app.nuts.NutsRepositorySecurityManager
```

⚙ Instance Methods

⚙ addUser(name)

```
NutsAddUserCommand addUser(String name)
```

null

⚙ checkAllowed(right, operationName)

```
NutsRepositorySecurityManager checkAllowed(String right, String operationName)
```

null null

⚙ checkCredentials(credentialsId, password)

check if the given `` password

```
by the Authentication Agent for ``
credentialsId
```

```
void checkCredentials(char[] credentialsId, char[] password)
```

credentialsId password

⚙ createCredentials(credentials, allowRetreive, credentialId)

store credentials in the agent's and return the credential id to store into the config. if credentialId is not null, the given credentialId will be updated and the credentialId is returned. The ` credentialsId` ,if present or returned, MUST be prefixed withAuthenticationAgent'd id and `:` character

```
char[] createCredentials(char[] credentials, boolean allowRetreive, char[] credentialId)
```

credential when true {@link #getCredentials(char[])} can be invoked over {@code credentialId} preferred credentialId, if null, a new one is created

 **findUsers()**

```
NutsUser[] findUsers()
```

 **getAuthenticationAgent(id)**

```
NutsAuthenticationAgent getAuthenticationAgent(String id)
```

null

 **getCredentials(credentialsId)**

get the credentials for the given id. The `credentialsId` === MUST** be prefixed with AuthenticationAgent'd id and ':' character

```
char[] getCredentials(char[] credentialsId)
```

credentials-id

 **getEffectiveUser(username)**

```
NutsUser getEffectiveUser(String username)
```

null

 **isAllowed(right)**

```
boolean isAllowed(String right)
```

null

⚙️ **removeCredentials(credentialsId)**

remove existing credentials with the given id The `credentialsId` === MUST** be prefixed with AuthenticationAgent'd id and ':' character

```
boolean removeCredentials(char[] credentialsId)
```

credentials-id

⚙️ **removeUser(name)**

```
NutsRemoveUserCommand removeUser(String name)
```

null

⚙️ **setAuthenticationAgent(authenticationAgent, options)**

```
NutsRepositorySecurityManager setAuthenticationAgent(String authenticationAgent,  
NutsUpdateOptions options)
```

null null

⚙️ **updateUser(name)**

```
NutsUpdateUserCommand updateUser(String name)
```

null

⚡️ **NutsUpdateCommand**

```
public net.vpc.app.nuts.NutsUpdateCommand
```

Instance Properties

📝 api

```
[read-write] NutsUpdateCommand api  
boolean isApi()  
NutsUpdateCommand setApi(enable)
```

📝 apiVersion

set target api version required for updating other artifacts

```
[read-write] NutsUpdateCommand apiVersion  
String getApiVersion()  
NutsUpdateCommand setApiVersion(value)
```

📝 args

```
[read-only] String[] args  
String[] getArgs()
```

📝 companions

```
[read-write] NutsUpdateCommand companions  
boolean isCompanions()  
NutsUpdateCommand setCompanions(updateCompanions)
```

📝 enableInstall

if true enable installing new artifacts when an update is request for non installed packages.

```
[read-write] NutsUpdateCommand enableInstall  
boolean isEnableInstall()  
NutsUpdateCommand setEnableInstall(enableInstall)
```

 extensions

```
[read-write] NutsUpdateCommand extensions
boolean isExtensions()
NutsUpdateCommand setExtensions(enable)
```

 ids

```
[read-only] NutsId[] ids
NutsId[] getIds()
```

 installed

```
[read-write] NutsUpdateCommand installed
boolean isInstalled()
NutsUpdateCommand setInstalled(enable)
```

 lockedIds

```
[read-only] NutsId[] lockedIds
NutsId[] getLockedIds()
```

 optional

when true include optional dependencies

```
[read-write] NutsUpdateCommand optional
boolean isOptional()
NutsUpdateCommand setOptional(includeOptional)
```

 result

execute update check (if not already performed) then return result

```
[read-only] NutsWorkspaceUpdateResult result  
NutsWorkspaceUpdateResult getResult()
```

📄 **resultCount**

```
[read-only] int resultCount  
int getResultCount()
```

📝 **runtime**

```
[read-write] NutsUpdateCommand runtime  
boolean isRuntime()  
NutsUpdateCommand setRuntime(enable)
```

🔗 **session**

update session

```
[write-only] NutsUpdateCommand session  
NutsUpdateCommand setSession(session)
```

⚙️ **Instance Methods**

⚙️ **addArg(arg)**

```
NutsUpdateCommand addArg(String arg)
```

null

⚙️ **addArgs(args)**

```
NutsUpdateCommand addArgs(Collection<String> args)
```

null

⚙️ **addArgs(args)**

```
NutsUpdateCommand addArgs(String args)
```

null

⚙️ **addId(id)**

```
NutsUpdateCommand addId(NutsId id)
```

null

⚙️ **addId(id)**

```
NutsUpdateCommand addId(String id)
```

null

⚙️ **addIds(ids)**

```
NutsUpdateCommand addIds(NutsId ids)
```

null

⚙️ **addIds(ids)**

```
NutsUpdateCommand addIds(String ids)
```

null

⚙️ **addLockedId(id)**

```
NutsUpdateCommand addLockedId(NutsId id)
```

null

⚙️ **addLockedId(id)**

```
NutsUpdateCommand addLockedId(String id)
```

null

⚙️ **addLockedIds(ids)**

```
NutsUpdateCommand addLockedIds(NutsId ids)
```

null

⚙️ **addLockedIds(ids)**

```
NutsUpdateCommand addLockedIds(String ids)
```

null

⚙️ **addScope(scope)**

```
NutsUpdateCommand addScope(NutsDependencyScope scope)
```

null

⚙️ **addScopes(scopes)**

```
NutsUpdateCommand addScopes(Collection<NutsDependencyScope> scopes)
```

null

⚙️ **addScopes(scopes)**

```
NutsUpdateCommand addScopes(NutsDependencyScope scopes)
```

null

 **checkUpdates()**

```
NutsUpdateCommand checkUpdates()
```

 **checkUpdates(applyUpdates)**

check for updates.

```
NutsUpdateCommand checkUpdates(boolean applyUpdates)
```

if true updates will be applied

 **clearArgs()**

```
NutsUpdateCommand clearArgs()
```

 **clearIds()**

```
NutsUpdateCommand clearIds()
```

 **clearLockedIds()**

```
NutsUpdateCommand clearLockedIds()
```

 **clearScopes()**

```
NutsUpdateCommand clearScopes()
```

⚙️ **companions()**

update workspace companion versions

```
NutsUpdateCommand companions()
```

⚙️ **companions(enable)**

```
NutsUpdateCommand companions(boolean enable)
```

null

⚙️ **configure(skipUnsupported, args)**

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsUpdateCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ **copySession()**

copy session

```
NutsUpdateCommand copySession()
```

⚙️ **installed()**

update installed artifacts

```
NutsUpdateCommand installed()
```

⚙️ `installed(enable)`

```
NutsUpdateCommand installed(boolean enable)
```

null

⚙️ `lockedId(id)`

```
NutsUpdateCommand lockedId(NutsId id)
```

null

⚙️ `lockedId(id)`

```
NutsUpdateCommand lockedId(String id)
```

null

⚙️ `lockedIds(id)`

```
NutsUpdateCommand lockedIds(NutsId id)
```

null

⚙️ `lockedIds(id)`

```
NutsUpdateCommand lockedIds(String id)
```

null

⚙️ `removeld(id)`

```
NutsUpdateCommand removeId(NutsId id)
```

null

⚙️ **removeld(id)**

```
NutsUpdateCommand removeId(String id)
```

null

⚙️ **run()**

execute the command and return this instance

```
NutsUpdateCommand run()
```

⚙️ **runtime()**

update workspace runtime version

```
NutsUpdateCommand runtime()
```

⚙️ **runtime(enable)**

```
NutsUpdateCommand runtime(boolean enable)
```

null

⚙️ **scope(scope)**

```
NutsUpdateCommand scope(NutsDependencyScope scope)
```

null

⚙️ **scopes(scopes)**

```
NutsUpdateCommand scopes(Collection<NutsDependencyScope> scopes)
```

null

 **scopes**(scopes)

```
NutsUpdateCommand scopes(NutsDependencyScope scopes)
```

null

 **setAll()**

update api, runtime, extensions, companions and all installed artifacts

```
NutsUpdateCommand setAll()
```

 **update()**

```
NutsUpdateCommand update()
```

 **workspace()**

update api, runtime, extensions and companions

```
NutsUpdateCommand workspace()
```

 **Permissions**

```
public static final net.vpc.app.nuts.Permissions
```

standard right keys for distinct operations in nuts.

Constant Fields

ADD_REPOSITORY

```
public static final String ADD_REPOSITORY = "add-repo"
```

ADMIN

```
public static final String ADMIN = "admin"
```

ALL

```
public static final Set<String> ALL = Collections.unmodifiableSet(new  
HashSet<>(Arrays.asList(FETCH_DESC, FETCH_CONTENT, SAVE, INSTALL, UPDATE,  
AUTO_INSTALL, UNINSTALL, EXEC, DEPLOY, UNDEPLOY, PUSH, ADD_REPOSITORY,  
REMOVE_REPOSITORY, SET_PASSWORD, ADMIN)))
```

AUTO_INSTALL

```
public static final String AUTO_INSTALL = "auto-install"
```

DEPLOY

```
public static final String DEPLOY = "deploy"
```

EXEC

```
public static final String EXEC = "exec"
```

FETCH_CONTENT

```
public static final String FETCH_CONTENT = "fetch-content"
```

 **FETCH_DESC**

```
public static final String FETCH_DESC = "fetch-desc"
```

 **INSTALL**

```
public static final String INSTALL = "install"
```

 **PUSH**

```
public static final String PUSH = "push"
```

 **REMOVE_REPOSITORY**

```
public static final String REMOVE_REPOSITORY = "remove-repo"
```

 **SAVE**

```
public static final String SAVE = "save"
```

 **SET_PASSWORD**

```
public static final String SET_PASSWORD = "set-password"
```

 **UNDEPLOY**

```
public static final String UNDEPLOY = "undeploy"
```

 **UNINSTALL**

```
public static final String UNINSTALL = "uninstall"
```

🔊 UPDATE

```
public static final String UPDATE = "update"
```

🔗 Constructors

🔗 Permissions()

private constructor

```
Permissions()
```

☕ QueryFaces

```
public static final net.vpc.app.nuts.QueryFaces
```

valid values for Query parameter "face"

🔊 Constant Fields

🔊 CONTENT

```
public static final String CONTENT = "content"
```

🔊 CONTENT_HASH

```
public static final String CONTENT_HASH = "content-hash"
```

🔊 DESCRIPTOR

```
public static final String DESCRIPTOR = "descriptor"
```

⌚ DESCRIPTOR_HASH

```
public static final String DESCRIPTOR_HASH = "descriptor-hash"
```

Constructors

QueryFaces()

private constructor

```
QueryFaces()
```

RepoTypes

```
public static final net.vpc.app.nuts.RepoTypes
```

Constant Fields

MAVEN

```
public static final String MAVEN = "maven"
```

NUTS

```
public static final String NUTS = "nuts"
```

NUTS_SERVER

```
public static final String NUTS_SERVER = "nuts-server"
```

Constructors

RepoTypes()

private constructor

```
RepoTypes()
```

SimpleConfirmDelete

```
private static net.vpc.app.nuts.SimpleConfirmDelete
```

Instance Fields

ignoreDeletion

```
private List<File> ignoreDeletion = new ArrayList<>()
```

Instance Properties

force

```
[read-write] boolean public force
private boolean force
public boolean isForce()
public void setForce(value)
```

Instance Methods

accept(directory)

```
boolean accept(File directory)
```

null

⚙ ignore(directory)

```
void ignore(File directory)
```

null

☕ Users

```
public static final net.vpc.app.nuts.Users
```

nuts standard user names

⌚ Constant Fields

⌚ ADMIN

```
public static final String ADMIN = "admin"
```

⌚ ANONYMOUS

```
public static final String ANONYMOUS = "anonymous"
```

Constructors

⌚ Users()

private constructor

```
Users()
```

☕ Versions

```
public static final net.vpc.app.nuts.Versions
```

version special names

Constant Fields

LATEST

```
public static final String LATEST = "LATEST"
```

RELEASE

```
public static final String RELEASE = "RELEASE"
```

Constructors

Versions()

private constructor

```
Versions()
```

Security

📝 NutsAddUserCommand

```
public net.vpc.app.nuts.NutsAddUserCommand
```

Command class for adding users to workspaces and repositories. All Command classes have a 'run' method to perform the operation.

🔗 Instance Properties

📝 ↗ credentials

```
[read-write] NutsAddUserCommand credentials  
char[] getCredentials()  
NutsAddUserCommand setCredentials(password)
```

📝 ↗ groups

group list defined by `#addGroup` , @link `#addGroups(String...)` } and @link `#addGroups(Collection)` }

```
[read-only] String[] groups  
String[] getGroups()
```

📝 ↗ permissions

return permissions

```
[read-only] String[] permissions  
String[] getPermissions()
```

📝 ↗ remoteCredentials

set remote credentials

```
[read-write] NutsAddUserCommand remoteCredentials  
char[] getRemoteCredentials()  
NutsAddUserCommand setRemoteCredentials(password)
```

remoteIdentity

set remote identity

```
[read-write] NutsAddUserCommand remoteIdentity  
String getRemoteIdentity()  
NutsAddUserCommand setRemoteIdentity(remoteIdentity)
```

session

update session

```
[write-only] NutsAddUserCommand session  
NutsAddUserCommand setSession(session)
```

username

set username

```
[read-write] NutsAddUserCommand username  
String getUsername()  
NutsAddUserCommand setUsername(username)
```

Instance Methods

addGroup(group)

add group named `group` to the specified user

```
NutsAddUserCommand addGroup(String group)
```

group name

⚙️ addGroups(groups)

add group list named `groups` to the specified user

```
NutsAddUserCommand addGroups(String groups)
```

group list

⚙️ addGroups(groups)

add group list named `groups` to the specified user

```
NutsAddUserCommand addGroups(Collection<String> groups)
```

group list

⚙️ addPermission(permission)

add permission named `permission` to the specified user

```
NutsAddUserCommand addPermission(String permission)
```

permission name from {@code NutsConstants.Permissions}

⚙️ addPermissions(permissions)

add permissions list named `permissions` to the specified user

```
NutsAddUserCommand addPermissions(String permissions)
```

group list

⚙️ addPermissions(permissions)

add permissions list named `permissions` to the specified user

```
NutsAddUserCommand addPermissions(Collection<String> permissions)
```

group list

⚙️ `configure(skipUnsupported, args)`

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsAddUserCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙️ `copySession()`

copy session

```
NutsAddUserCommand copySession()
```

⚙️ `removeGroups(groups)`

remove group

```
NutsAddUserCommand removeGroups(String groups)
```

new value

⚙️ `removeGroups(groups)`

remove groups

```
NutsAddUserCommand removeGroups(Collection<String> groups)
```

groups to remove

⚙️ removePermissions(permissions)

remove permissions

```
NutsAddUserCommand removePermissions(String permissions)
```

permission to remove

⚙️ removePermissions(permissions)

remove permissions

```
NutsAddUserCommand removePermissions(Collection<String> permissions)
```

permissions to remove

⚙️ run()

execute the command and return this instance

```
NutsAddUserCommand run()
```

☕ NutsAuthenticationAgent

```
public net.vpc.app.nuts.NutsAuthenticationAgent
```

an Authentication Agent is responsible of storing and retrieving credentials in external repository (password manager, kwallet, keypads, gnome-keyring...). And Id of the stored password is then saved as plain text in nuts config file. Criteria type is a string representing authentication agent id

⌚ Instance Properties

📄⌚ id

agent id;

```
[read-only] String id
String getId()
```

⚙ Instance Methods

⚙ checkCredentials(credentialsId, password, envProvider)

check if the given `` password

```
by the Authentication Agent for ``
credentialsId
```

```
void checkCredentials(char[] credentialsId, char[] password, Map<String, String>
envProvider)
```

credentialsId password environment provider, nullable

⚙ createCredentials(credentials, allowRetrieve, credentialId, envProvider)

store credentials in the agent's and return the credential id to store into the config. if credentialId is not null, the given credentialId will be updated and the credentialId is returned. The `credentialsId` ,if present or returned, MUST be prefixed withAuthenticationAgent'd id and ':' character

```
char[] createCredentials(char[] credentials, boolean allowRetrieve, char[]
credentialId, Map<String, String> envProvider)
```

credential when true {@link #getCredentials(char[], Map)} }can be invoked over {@code credentialId} preferred credentialId, if null, a new one is created environment provider, nullable

⚙ getCredentials(credentialsId, envProvider)

get the credentials for the given id. The ` credentialsId` === MUST** be prefixed with AuthenticationAgent'd id and ':' character

```
char[] getCredentials(char[] credentialsId, Map<String, String> envProvider)
```

credentials-id environment provider, nullable

removeCredentials(credentialsId, envProvider)

remove existing credentials with the given id The `credentialsId` **==> MUST**** be prefixed with AuthenticationAgent'd id and ':' character

```
boolean removeCredentials(char[] credentialsId, Map<String, String> envProvider)
```

credentials-id environment provider, nullable

NutsRemoveUserCommand

```
public net.vpc.app.nuts.NutsRemoveUserCommand
```

Remove User Command

Instance Properties

session

update session

```
[write-only] NutsRemoveUserCommand session
NutsRemoveUserCommand setSession(session)
```

username

set username of user to remove

```
[read-write] NutsRemoveUserCommand username
String getUsername()
NutsRemoveUserCommand setUsername(username)
```

⚙ Instance Methods

⚙ configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsRemoveUserCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙ copySession()

copy session

```
NutsRemoveUserCommand copySession()
```

⚙ run()

execute the command and return this instance

```
NutsRemoveUserCommand run()
```

⚙ username(username)

set username of user to remove

```
NutsRemoveUserCommand username(String username)
```

user name

☕ NutsUpdateUserCommand

```
public net.vpc.app.nuts.NutsUpdateUserCommand
```

Instance Properties

addGroups

```
[read-only] String[] addGroups  
String[] getAddGroups()
```

addPermissions

```
[read-only] String[] addPermissions  
String[] getAddPermissions()
```

credentials

```
[read-write] NutsUpdateUserCommand credentials  
char[] getCredentials()  
NutsUpdateUserCommand setCredentials(password)
```

oldCredentials

```
[read-write] NutsUpdateUserCommand oldCredentials  
char[] getOldCredentials()  
NutsUpdateUserCommand setOldCredentials(oldCredentials)
```

remoteCredentials

```
[read-write] NutsUpdateUserCommand remoteCredentials  
char[] getRemoteCredentials()  
NutsUpdateUserCommand setRemoteCredentials(password)
```

remoteIdentity

```
[read-write] NutsUpdateUserCommand remoteIdentity  
String getRemoteIdentity()  
NutsUpdateUserCommand setRemoteIdentity(remoteIdentity)
```

[removeGroups](#)

```
[read-only] String[] removeGroups  
String[] getRemoveGroups()
```

[removePermissions](#)

```
[read-only] String[] removePermissions  
String[] getRemovePermissions()
```

[resetGroups](#)

```
[read-write] NutsUpdateUserCommand resetGroups  
boolean isResetGroups()  
NutsUpdateUserCommand setResetGroups(resetGroups)
```

[resetPermissions](#)

```
[read-write] NutsUpdateUserCommand resetPermissions  
boolean isResetPermissions()  
NutsUpdateUserCommand setResetPermissions(resetPermissions)
```

[session](#)

update session

```
[write-only] NutsUpdateUserCommand session  
NutsUpdateUserCommand setSession(session)
```

 `username`

```
[read-write] NutsUpdateUserCommand username
String getUsername()
NutsUpdateUserCommand setUsername(login)
```

Instance Methods

`addGroup(group)`

```
NutsUpdateUserCommand addGroup(String group)
```

null

`addGroups(groups)`

```
NutsUpdateUserCommand addGroups(String groups)
```

null

`addGroups(groups)`

```
NutsUpdateUserCommand addGroups(Collection<String> groups)
```

null

`addPermission(permission)`

```
NutsUpdateUserCommand addPermission(String permission)
```

null

`addPermissions(permissions)`

```
NutsUpdateUserCommand addPermissions(String permissions)
```

null

⚙ **addPermissions**(permissions)

```
NutsUpdateUserCommand addPermissions(Collection<String> permissions)
```

null

⚙ **configure**(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsUpdateUserCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

⚙ **copySession()**

copy session

```
NutsUpdateUserCommand copySession()
```

⚙ **credentials**(password)

```
NutsUpdateUserCommand credentials(char[] password)
```

null

⚙ **oldCredentials**(password)

```
NutsUpdateUserCommand oldCredentials(char[] password)
```

null

⚙️ **remoteCredentials(password)**

```
NutsUpdateUserCommand remoteCredentials(char[] password)
```

null

⚙️ **remoteIdentity(remoteIdentity)**

```
NutsUpdateUserCommand remoteIdentity(String remoteIdentity)
```

null

⚙️ **removeGroup(group)**

```
NutsUpdateUserCommand removeGroup(String group)
```

null

⚙️ **removeGroups(groups)**

```
NutsUpdateUserCommand removeGroups(String groups)
```

null

⚙️ **removeGroups(groups)**

```
NutsUpdateUserCommand removeGroups(Collection<String> groups)
```

null

⚙️ **removePermission(permission)**

```
NutsUpdateUserCommand removePermission(String permission)
```

null

⚙️ `removePermissions(permissions)`

```
NutsUpdateUserCommand removePermissions(String permissions)
```

null

⚙️ `removePermissions(permissions)`

```
NutsUpdateUserCommand removePermissions(Collection<String> permissions)
```

null

⚙️ `resetGroups()`

```
NutsUpdateUserCommand resetGroups()
```

⚙️ `resetGroups(resetGroups)`

```
NutsUpdateUserCommand resetGroups(boolean resetGroups)
```

null

⚙️ `resetPermissions()`

```
NutsUpdateUserCommand resetPermissions()
```

⚙️ `resetPermissions(resetPermissions)`

```
NutsUpdateUserCommand resetPermissions(boolean resetPermissions)
```

null

⚙️ **run()**

execute the command and return this instance

NutsUpdateUserCommand **run()**

⚙️ **undoAddGroup(group)**

NutsUpdateUserCommand **undoAddGroup(String group)**

null

⚙️ **undoAddGroups(groups)**

NutsUpdateUserCommand **undoAddGroups(String groups)**

null

⚙️ **undoAddGroups(groups)**

NutsUpdateUserCommand **undoAddGroups(Collection<String> groups)**

null

⚙️ **undoAddPermission(permissions)**

NutsUpdateUserCommand **undoAddPermission(String permissions)**

null

⚙️ **undoAddPermissions(permissions)**

```
NutsUpdateUserCommand undoAddPermissions(String permissions)
```

null

⚙ **undoAddPermissions**(permissions)

```
NutsUpdateUserCommand undoAddPermissions(Collection<String> permissions)
```

null

⚙ **undoRemoveGroups**(groups)

```
NutsUpdateUserCommand undoRemoveGroups(String groups)
```

null

⚙ **undoRemoveGroups**(groups)

```
NutsUpdateUserCommand undoRemoveGroups(Collection<String> groups)
```

null

⚙ **undoRemovePermissions**(permissions)

```
NutsUpdateUserCommand undoRemovePermissions(String permissions)
```

null

⚙ **undoRemovePermissions**(permissions)

```
NutsUpdateUserCommand undoRemovePermissions(Collection<String> permissions)
```

null

⚙️ username(login)

```
NutsUpdateUserCommand username(String login)
```

null

☕ NutsWorkspaceSecurityManager

```
public net.vpc.app.nuts.NutsWorkspaceSecurityManager
```

Workspace Security configuration manager

⌚ Instance Properties

📋⌚ admin

return true if current user has admin privileges

```
[read-only] boolean admin  
boolean isAdmin()
```

📋⌚ currentLoginStack

current user stack. this is useful when login with multiple user identities.

```
[read-only] String[] currentLoginStack  
String[] getCurrentLoginStack()
```

📋⌚ currentUsername

current user

```
[read-only] String currentUsername  
String getCurrentUsername()
```

secure

return true if workspace is running secure mode

```
[read-only] boolean secure  
boolean isSecure()
```

Instance Methods

addUser(name)

create a User Create command. No user will be added when simply calling this method. You must fill in command parameters then call ` NutsAddUserCommand#run()` .

```
NutsAddUserCommand addUser(String name)
```

user name

checkAllowed(permission, operationName)

check if allowed and throw a Security exception if not.

```
NutsWorkspaceSecurityManager checkAllowed(String permission, String  
operationName)
```

permission name. see {@code NutsConstants.Rights } class operation name

checkCredentials(credentialsId, password)

check if the given `` password

```
by the Authentication Agent for '''  
credentialsId
```

```
void checkCredentials(char[] credentialsId, char[] password)
```

credentialsId password

⚙️ `createCredentials(credentials, allowRetrieve, credentialId)`

store credentials in the agent's and return the credential id to store into the config. if credentialId is not null, the given credentialId will be updated and the credentialId is returned. The `credentialsId` ,if present or returned, MUST be prefixed withAuthenticationAgent'd id and ':' character

```
char[] createCredentials(char[] credentials, boolean allowRetrieve, char[] credentialId)
```

credential when true {@link #getCredentials(char[])} can beinvoked over {@code credentialId} preferred credentialId, if null, a new one is created

⚙️ `currentLoginStack()`

equivalent to `#getCurrentLoginStack()` .

```
String[] currentLoginStack()
```

⚙️ `currentUser()`

equivalent to `#getCurrentUsername()` .

```
String currentUser()
```

⚙️ `findUser(username)`

find user with the given name or null.

```
NutsUser findUser(String username)
```

user name

⚙ **findUsers()**

find all registered users

```
NutsUser[] findUsers()
```

⚙ **getAuthenticationAgent(authenticationAgentId)**

get authentication agent with id `authenticationAgentId` .if is blank, return default authentication agent

```
NutsAuthenticationAgent getAuthenticationAgent(String authenticationAgentId)
```

agent id

⚙ **getCredentials(credentialsId)**

get the credentials for the given id. The `credentialsId` === MUST** be prefixed with AuthenticationAgent'd id and ':' character

```
char[] getCredentials(char[] credentialsId)
```

credentials-id

⚙ **isAllowed(permission)**

return true if permission is valid and allowed for the current user.

```
boolean isAllowed(String permission)
```

permission name. see {@code NutsConstants.Rights } class

⚙ **login(handler)**

impersonate user and log as a distinct user with the given credentials and stack user name so that it can be retrieved using {@code getCurrentLoginStack()}.

```
NutsWorkspaceSecurityManager login(CallbackHandler handler)
```

security handler

⚙️ login(username, password)

impersonate user and log as a distinct user with the given credentials.

```
NutsWorkspaceSecurityManager login(String username, char[] password)
```

user name user password

⚙️ logout()

log out from last logged in user (if any) and pop out from user name stack.

```
NutsWorkspaceSecurityManager logout()
```

⚙️ removeCredentials(credentialsId)

remove existing credentials with the given id The `credentialsId` === MUST** be prefixed with AuthenticationAgent'd id and ':' character

```
boolean removeCredentials(char[] credentialsId)
```

credentials-id

⚙️ removeUser(name)

create a Remove Create command. No user will be removed when simply calling this method. You must fill in command parameters then call ` NutsRemoveUserCommand#run()` .

```
NutsRemoveUserCommand removeUser(String name)
```

user name

⚙️ **setAuthenticationAgent(authenticationAgentId, options)**

update default authentication agent.

```
NutsWorkspaceSecurityManager setAuthenticationAgent(String authenticationAgentId,  
NutsUpdateOptions options)
```

authentication agent id update options

⚙️ **setSecureMode(secure, adminPassword, options)**

switch from/to secure mode. when secure mode is disabled, no authorizations are checked against.

```
boolean setSecureMode(boolean secure, char[] adminPassword, NutsUpdateOptions  
options)
```

true if secure mode password for admin user update options

⚙️ **updateUser(name)**

create a Update Create command. No user will be updated when simply calling this method. You must fill in command parameters then call `NutsUpdateUserCommand#run()` .

```
NutsUpdateUserCommand updateUser(String name)
```

user name

SPI Base

☕ NutsBootClassLoader

```
net.vpc.app.nuts.NutsBootClassLoader
```

Simple Implementation of Nuts BootClassLoader

Constructors

⌚ NutsBootClassLoader(urls, parent)

default constructor

```
NutsBootClassLoader(URL[] urls, ClassLoader parent)
```

urls parent class loader

☕ NutsBootWorkspace

```
public final net.vpc.app.nuts.NutsBootWorkspace
```

NutsBootWorkspace is responsible of loading initial nuts-core.jar and its dependencies and for creating workspaces using the method `#openWorkspace()`. NutsBootWorkspace is also responsible of managing local jar cache folder located at ~/.cache/nuts/default-workspace/boot

Default Bootstrap implementation. This class is responsible of loading initial nuts-core.jar and its dependencies and for creating workspaces using the method `#openWorkspace()`.

⌚ Constant Fields

⌚ DELETE_FOLDERS_HEADER

```
private static final String DELETE_FOLDERS_HEADER = "ATTENTION ! You are about  
to delete nuts workspace files."
```

[Instance Fields](#)

[LOG](#)

```
private PrivateNutsLog LOG = new PrivateNutsLog()
```

[creationTime](#)

```
private final long creationTime = System.currentTimeMillis()
```

[newInstanceRequirements](#)

```
private int newInstanceRequirements
```

[parsedBootRepositories](#)

```
private Set<String> parsedBootRepositories
```

[pathExpansionConverter](#)

```

private final Function<String, String> pathExpansionConverter = new
Function<String, String>() {

    @Override
    public String apply(String from) {
        switch(from) {
            case "workspace":
                return workspaceInformation.getWorkspaceLocation();
            case "user.home":
                return System.getProperty("user.home");
            case "home.apps":
            case "home.config":
            case "home.lib":
            case "home.temp":
            case "home.var":
            case "home.cache":
            case "home.run":
            case "home.log":
                return getHome(NutsStoreLocation.valueOf(from.substring("home."
.length()).toUpperCase()));
            case "apps":
            case "config":
            case "lib":
            case "cache":
            case "run":
            case "temp":
            case "log":
            case "var":
                {
                    Map<String, String> s =
workspaceInformation.getStoreLocations();
                    if (s == null) {
                        return " + from + ";
                    }
                    return s.get(from);
                }
            }
        return " + from + ";
    }
}

```

[preparedWorkspace](#)

```
private boolean preparedWorkspace
```

🔗 workspaceInformation

```
private PrivateNutsWorkspaceInitInformation workspaceInformation
```

🔗 Constructors

🔗 NutsBootWorkspace(args)

```
NutsBootWorkspace(String args)
```

null

🔗 NutsBootWorkspace(options)

```
NutsBootWorkspace(NutsWorkspaceOptions options)
```

null

🔗 Instance Properties

📄 ↗ contextClassLoader

```
[read-only] protected ClassLoader contextClassLoader  
protected ClassLoader getContextClassLoader()
```

📄 ↗ contextClassLoaderSupplier

```
[read-only] private Supplier<ClassLoader> contextClassLoaderSupplier  
private Supplier<ClassLoader> contextClassLoaderSupplier  
private Supplier<ClassLoader> getContextClassLoaderSupplier()
```

📄 loadFromCache

```
[read-only] private boolean loadFromCache  
private boolean isLoadFromCache()
```

📄 options

```
[read-only] public NutsWorkspaceOptions options  
private NutsWorkspaceOptions options  
public NutsWorkspaceOptions getOptions()
```

📄 workspaceRunModeString

```
[read-only] private String workspaceRunModeString  
private String getWorkspaceRunModeString()
```

⚙️ Instance Methods

⚙️ checkRequirements(unsatisfiedOnly)

build and return unsatisfied requirements

```
int checkRequirements(boolean unsatisfiedOnly)
```

when true return requirements for new instance

⚙️ createProcessBuilder()

```
ProcessBuilder createProcessBuilder()
```

⚙️ createProcessCommandLine()

```
String[] createProcessCommandLine()
```

⚙️ **deleteStoreLocations(includeRoot, locations)**

```
void deleteStoreLocations(boolean includeRoot, Object locations)
```

true if include root of type NutsStoreLocation, Path or File

⚙️ **expandPath(path, base)**

```
String expandPath(String path, String base)
```

null null

⚙️ **fallbackInstallActionUnavailable(message)**

```
void fallbackInstallActionUnavailable(String message)
```

null

⚙️ **getBootCacheFile(path, repository, cacheFolder, useCache)**

```
File getBootCacheFile(String path, String repository, String cacheFolder, boolean useCache)
```

null null null null

⚙️ **getBootCacheFile(vid, fileName, repositories, cacheFolder, useCache)**

```
File getBootCacheFile(PrivateNutsId vid, String fileName, String[] repositories, String cacheFolder, boolean useCache)
```

null null null null null

⚙️ **getBootCacheJar(id, repositories, cacheFolder, useCache, name)**

```
File getBootCacheJar(String id, String[] repositories, String cacheFolder,  
boolean useCache, String name)
```

null null null null null

⚙ **getFileName**(id, ext)

```
String getFileName(PrivateNutsId id, String ext)
```

null null

⚙ **getHome**(storeFolder)

```
String getHome(NutsStoreLocation storeFolder)
```

null

⚙ **getPathFile**(id, name)

```
String getPathFile(PrivateNutsId id, String name)
```

null null

⚙ **getRequirementsHelpString**(unsatisfiedOnly)

return a string representing unsatisfied contrains

```
String getRequirementsHelpString(boolean unsatisfiedOnly)
```

when true return requirements for new instance

⚙ **getStoreLocationPath**(value)

```
String getStoreLocationPath(NutsStoreLocation value)
```

null

⚙️ **hasUnsatisfiedRequirements()**

```
boolean hasUnsatisfiedRequirements()
```

⚙️ **isLoadedClassPath(file)**

```
boolean isLoadedClassPath(File file)
```

null

⚙️ **openWorkspace()**

```
NutsWorkspace openWorkspace()
```

⚙️ **prepareWorkspace()**

```
boolean prepareWorkspace()
```

⚙️ **resolveBootRepositories()**

```
Collection<String> resolveBootRepositories()
```

⚙️ **resolveClassWorldURLs(list)**

```
URL[] resolveClassWorldURLs(Collection<File> list)
```

null

⚙️ **runWorkspace()**

```
void runWorkspace()
```

 **showError(actualBootConfig, workspace, bootClassWorldURLs, extraMessage)**

```
void showError(PrivateNutsWorkspaceInitInformation actualBootConfig, String workspace, URL[] bootClassWorldURLs, String extraMessage)
```

null null null null

 **startNewProcess()**

```
int startNewProcess()
```

 NutsBootWorkspaceFactory

```
public net.vpc.app.nuts.NutsBootWorkspaceFactory
```

Class responsible of creating and initializing Workspace Created by vpc on 1/5/17.

 **Instance Methods**

 **createWorkspace(options)**

create workspace with the given options

```
NutsWorkspace createWorkspace(NutsWorkspaceInitInformation options)
```

boot init options

 **getBootSupportLevel(options)**

when multiple factories are available, the best one is selected according to the maximum value of `getBootSupportLevel(options)` .Note that default value (for the reference implementation) is `NutsComponent.DEFAULT_SUPPORT` .Any value less or equal to zero is ignored (and the factory is discarded)

```
int getBootSupportLevel(NutsWorkspaceOptions options)
```

command line options

✍ NutsCommandAliasFactoryConfig

```
public net.vpc.app.nuts.NutsCommandAliasFactoryConfig
```

Command Alias Factory Definition Config

✍ Constant Fields

✍ serialVersionUID

```
private static final long serialVersionUID = 1
```

✍ Instance Properties

✍ factoryId

Factory id (unique identifier in the workspace)

```
[read-write] String public factoryId  
private String factoryId  
public String getFactoryId()  
public NutsCommandAliasFactoryConfig setFactoryId(value)
```

✍ factoryType

Factory Type

```
[read-write] String public factoryType  
private String factoryType  
public String getFactoryType()  
public NutsCommandAliasFactoryConfig setFactoryType(value)
```

parameters

factory parameters

```
[read-write] Map<String, String> public parameters
private Map<String, String> parameters
public Map<String, String> getParameters()
public NutsCommandAliasFactoryConfig setParameters(value)
```

priority

priority (the higher the better)

```
[read-write] int public priority
private int priority
public int getPriority()
public NutsCommandAliasFactoryConfig setPriority(value)
```

Instance Methods

equals(o)

```
boolean equals(Object o)
```

null

hashCode()

```
int hashCode()
```

toString()

```
String toString()
```

☕ NutsComponent

```
public net.vpc.app.nuts.NutsComponent
```

Top Level extension Point in Nuts. Extension mechanism in nuts is based on a factory that selects the best implementation for a given predefined interface (named Extension Point). Such interfaces must extend this `NutsComponent` interface. Implementations must implement these extension points by providing their best support level (when method `#getSupportLevel(net.vpc.app.nuts.NutsSupportLevelContext)` is invoked). Only implementations with positive support level are considered. Implementations with higher support level are selected first.

⌚ Instance Fields

⌚ CUSTOM_SUPPORT

```
int CUSTOM_SUPPORT = 1000
```

⌚ DEFAULT_SUPPORT

```
int DEFAULT_SUPPORT = 10
```

⌚ NO_SUPPORT

```
int NO_SUPPORT = -1
```

⚙️ Instance Methods

⚙️ getSupportLevel(context)

evaluate support level (how much this instance should be considered convenient, acceptable) for the given arguments (provided in context).

```
int getSupportLevel(NutsSupportLevelContext<CriteriaType> context)
```

evaluation context

⌚ NutsDefaultSupportLevelContext

```
public net.vpc.app.nuts.NutsDefaultSupportLevelContext
```

Default and dummy NutsSupportLevelContext implementation

⌚ Instance Fields

⌚ ws

```
private final NutsWorkspace ws
```

⌚ Constructors

⌚ NutsDefaultSupportLevelContext(ws, constraints)

default constructor

```
NutsDefaultSupportLevelContext(NutsWorkspace ws, T constraints)
```

workspace constraints

⌚ Instance Properties

⌚constraints

```
[read-only] public T constraints
private final T constraints
public T getConstraints()
```

workspace

```
[read-only] public NutsWorkspace workspace  
public NutsWorkspace getWorkspace()
```

Instance Methods

equals(o)

```
boolean equals(Object o)
```

null

hashCode()

```
int hashCode()
```

toString()

```
String toString()
```

NutsDeployRepositoryCommand

```
public net.vpc.app.nuts.NutsDeployRepositoryCommand
```

Repository Deploy command provided by Repository and used by Workspace. This class is part of Nuts SPI and is not to be used by end users.

Instance Properties

content

set content to deploy

```
[read-write] NutsDeployRepositoryCommand content  
Object getContent()  
NutsDeployRepositoryCommand setContent(content)
```

descriptor

set descriptor to deploy

```
[read-write] NutsDeployRepositoryCommand descriptor  
NutsDescriptor getDescriptor()  
NutsDeployRepositoryCommand setDescriptor(descriptor)
```

id

set id to deploy

```
[read-write] NutsDeployRepositoryCommand id  
NutsId getId()  
NutsDeployRepositoryCommand setId(id)
```

session

session

```
[write-only] NutsDeployRepositoryCommand session  
NutsDeployRepositoryCommand setSession(session)
```

Instance Methods

run()

run deploy command

```
NutsDeployRepositoryCommand run()
```

☕ NutsDescriptorContentParserComponent

```
public net.vpc.app.nuts.NutsDescriptorContentParserComponent
```

Content parser component is responsible of resolving a Nuts descriptor form a content file

⚙ Instance Methods

⚙ parse(parserContext)

parse content and return a valid NutsDescriptor or null if not supported.

```
NutsDescriptor parse(NutsDescriptorContentParserContext parserContext)
```

context

☕ NutsDescriptorContentParserContext

```
public net.vpc.app.nuts.NutsDescriptorContentParserContext
```

context holding useful information for
NutsDescriptorContentParserComponent#parse(NutsDescriptorContentParserContext)`
==== ⓘ
Instance Properties

📄 ⓘ fileExtension

content file extension or null. At least one of file extension or file mime-type is provided.

```
[read-only] String fileExtension  
String getFileExtension()
```

📄 ⓘ fullStream

content stream

```
[read-only] InputStream fullStream
InputStream getFullStream()
```

headStream

return content header stream. if the content size is less than 1Mb, then all the content is returned. If not, at least 1Mb is returned.

```
[read-only] InputStream headStream
InputStream getHeadStream()
```

contentType

content mime-type or null. At least one of file extension or file mime-type is provided.

```
[read-only] String mimeType
String getMimeType()
```

name

content name (mostly content file name)

```
[read-only] String name
String getName()
```

parseOptions

command line options that can be parsed to configure parsing options. A good example of it is the --all-mains option that can be passed as executor option which will be catched by parser to force resolution of all main classes even though a Main-Class attribute is visited in the MANIFEST.MF file. This array may continue any non supported options. They should be discarded by the parser.

```
[read-only] String[] parseOptions
String[] getParseOptions()
```

session

return session

```
[read-only] NutsSession session  
NutsSession getSession()
```

workspace

return workspace

```
[read-only] NutsWorkspace workspace  
NutsWorkspace getWorkspace()
```

NutsExecutorComponent

```
public net.vpc.app.nuts.NutsExecutorComponent
```

An Executor Component is responsible of "executing" a nuts component (package) Created by vpc on 1/7/17.

Instance Properties

id

artifact id

```
[read-only] NutsId id  
NutsId getId()
```

Instance Methods

dryExec(executionContext)

performs a dry execution (simulation) avoiding any side effect and issuing trace to standard output in order to log simulation workflow.

```
void dryExec(NutsExecutionContext executionContext)
```

executionContext

 **exec(executionContext)**

execute the artifact

```
void exec(NutsExecutionContext executionContext)
```

executionContext

 **NutsFetchContentRepositoryCommand**

```
public net.vpc.app.nuts.NutsFetchContentRepositoryCommand
```

Repository command bound to FetchCommand used to fetch an artifact content from a specific repository.

 **Instance Properties**

 **descriptor**

set descriptor to fetch.

```
[read-write] NutsFetchContentRepositoryCommand descriptor
NutsDescriptor getDescriptor()
NutsFetchContentRepositoryCommand setDescriptor(descriptor)
```

 **fetchMode**

fetchMode

```
[read-write] NutsFetchContentRepositoryCommand fetchMode
NutsFetchMode getFetchMode()
NutsFetchContentRepositoryCommand setFetchMode(fetchMode)
```

id

set id to fetch.

```
[read-write] NutsFetchContentRepositoryCommand id  
NutsId getId()  
NutsFetchContentRepositoryCommand setId(id)
```

localPath

set localPath to store to.

```
[read-write] NutsFetchContentRepositoryCommand localPath  
Path getPath()  
NutsFetchContentRepositoryCommand setLocalPath(localPath)
```

result

return fetch result. if the command is not yet executed, it will be executed first.

```
[read-only] NutsContent result  
NutsContent getResult()
```

session

set current session.

```
[write-only] NutsFetchContentRepositoryCommand session  
NutsFetchContentRepositoryCommand setSession(session)
```

Instance Methods

run()

preform command. Should be called after setting all parameters. Result is retrievable with `#getResult()` .

NutsFetchContentRepositoryCommand [run\(\)](#)

⌚ [NutsFetchDescriptorRepositoryCommand](#)

```
public net.vpc.app.nuts.NutsFetchDescriptorRepositoryCommand
```

Repository command used to fetch an artifact descriptor from a specific repository.

⌚ [Instance Properties](#)

📝⌚ [fetchMode](#)

fetchMode

```
[read-write] NutsFetchDescriptorRepositoryCommand fetchMode  
  NutsFetchMode getFetchMode\(\)  
  NutsFetchDescriptorRepositoryCommand setFetchMode(fetchMode)
```

📝⌚ [id](#)

set id to fetch

```
[read-write] NutsFetchDescriptorRepositoryCommand id  
  NutsId getId\(\)  
  NutsFetchDescriptorRepositoryCommand setId(id)
```

📝⌚ [result](#)

return fetch result. if the command is not yet executed, it will be executed first.

```
[read-only] NutsDescriptor result  
  NutsDescriptor getResult\(\)
```

session

```
[write-only] NutsFetchDescriptorRepositoryCommand session  
NutsFetchDescriptorRepositoryCommand setSession(session)
```

Instance Methods

run()

preform command. Should be called after setting all parameters. Result is retrievable with `#getResult()`.

```
NutsFetchDescriptorRepositoryCommand run()
```

NutsInstallerComponent

```
public net.vpc.app.nuts.NutsInstallerComponent
```

Component responsible of installing other artifacts.

Instance Methods

install(executionContext)

install artifact

```
void install(NutsExecutionContext executionContext)
```

execution context

uninstall(executionContext, deleteData)

uninstall artifact

```
void uninstall(NutsExecutionContext executionContext, boolean deleteData)
```

execution context delete data after uninstall

update(executionContext)

update artifact

```
void update(NutsExecutionContext executionContext)
```

execution context

NutsPushRepositoryCommand

```
public net.vpc.app.nuts.NutsPushRepositoryCommand
```

Push Command

Instance Properties

args

args args to push

```
[read-write] NutsPushRepositoryCommand args
String[] getArgs()
NutsPushRepositoryCommand setArgs(args)
```

id

set id to push.

```
[read-write] NutsPushRepositoryCommand id
NutsId getId()
NutsPushRepositoryCommand setId(id)
```

offline

local only (installed or not)

```
[read-write] NutsPushRepositoryCommand offline  
boolean isOffline()  
NutsPushRepositoryCommand setOffline(offline)
```

repository

repository to push from

```
[read-write] NutsPushRepositoryCommand repository  
String getRepository()  
NutsPushRepositoryCommand setRepository(repository)
```

session

set session

```
[write-only] NutsPushRepositoryCommand session  
NutsPushRepositoryCommand setSession(session)
```

Instance Methods

run()

run this command and return `this` instance

```
NutsPushRepositoryCommand run()
```

NutsRepositoryCommand

```
public net.vpc.app.nuts.NutsRepositoryCommand
```

Root class for all Repository commands.

Instance Properties

session

set session

```
[read-write] NutsRepositoryCommand session  
NutsSession getSession()  
NutsRepositoryCommand setSession(session)
```

Instance Methods

configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsRepositoryCommand configure(boolean skipUnsupported, String args)
```

when true, all unsupported options are skipped argument to configure with

run()

run this command and return `this` instance

```
NutsRepositoryCommand run()
```

NutsRepositoryFactoryComponent

```
public net.vpc.app.nuts.NutsRepositoryFactoryComponent
```

Created by vpc on 1/15/17.

Instance Methods

⚙️ `create(options, workspace, parentRepository)`

```
NutsRepository create(NutsAddRepositoryOptions options, NutsWorkspace workspace,  
NutsRepository parentRepository)
```

null null null

⚙️ `getDefaultRepositories(workspace)`

```
NutsRepositoryDefinition[] getDefaultRepositories(NutsWorkspace workspace)
```

null

〝 `NutsRepositoryUndeployCommand`

```
public net.vpc.app.nuts.NutsRepositoryUndeployCommand
```

⌚ Instance Properties

📝⌚ `id`

```
[read-write] NutsRepositoryUndeployCommand id  
NutsId getId()  
NutsRepositoryUndeployCommand setId(id)
```

📝⌚ `offline`

```
[read-write] NutsRepositoryUndeployCommand offline  
boolean isOffline()  
NutsRepositoryUndeployCommand setOffline(offline)
```

📝⌚ `repository`

```
[read-write] NutsRepositoryUndeployCommand repository
String getRepository()
NutsRepositoryUndeployCommand setRepository(repository)
```

session

```
[write-only] NutsRepositoryUndeployCommand session
NutsRepositoryUndeployCommand setSession(session)
```

transitive

```
[read-write] NutsRepositoryUndeployCommand transitive
boolean isTransitive()
NutsRepositoryUndeployCommand setTransitive(transitive)
```

Instance Methods

run()

run this command and return `this` instance

```
NutsRepositoryUndeployCommand run()
```

NutsSearchRepositoryCommand

```
public net.vpc.app.nuts.NutsSearchRepositoryCommand
```

Instance Properties

fetchMode

fetchMode

```
[read-write] NutsSearchRepositoryCommand fetchMode
  NutsFetchMode getFetchMode()
  NutsSearchRepositoryCommand setFetchMode(fetchMode)
```

filter

```
[read-write] NutsSearchRepositoryCommand filter
  NutsIdFilter getFilter()
  NutsSearchRepositoryCommand setFilter(filter)
```

result

this method should return immediately and returns valid iterator. visiting iterator may be blocking but not this method call. If `run()` method has not been called yet, it will be called.

```
[read-only] Iterator<NutsId> result
  Iterator<NutsId> getResult()
```

session

```
[write-only] NutsSearchRepositoryCommand session
  NutsSearchRepositoryCommand setSession(session)
```

Instance Methods

run()

this method should return immediately after initializing a valid iterator to be retrieved by `getResult()`

```
NutsSearchRepositoryCommand run()
```

NutsSearchVersionsRepositoryCommand

```
public net.vpc.app.nuts.NutsSearchVersionsRepositoryCommand
```

Instance Properties

`fetchMode`

fetchMode

```
[read-write] NutsSearchVersionsRepositoryCommand fetchMode  
NutsFetchMode getFetchMode()  
NutsSearchVersionsRepositoryCommand setFetchMode(fetchMode)
```

`filter`

```
[read-write] NutsSearchVersionsRepositoryCommand filter  
NutsIdFilter getFilter()  
NutsSearchVersionsRepositoryCommand setFilter(filter)
```

`id`

```
[read-write] NutsSearchVersionsRepositoryCommand id  
NutsId getId()  
NutsSearchVersionsRepositoryCommand setId(id)
```

`result`

```
[read-only] Iterator<NutsId> result  
Iterator<NutsId> getResult()
```

`session`

```
[write-only] NutsSearchVersionsRepositoryCommand session  
NutsSearchVersionsRepositoryCommand setSession(session)
```

⚙ Instance Methods

⚙ run()

run this command and return `this` instance

```
NutsSearchVersionsRepositoryCommand run()
```

☕ NutsServiceLoader

```
public net.vpc.app.nuts.NutsServiceLoader
```

Component service class loader.

⚙ Instance Methods

⚙ loadAll(criteria)

load all NutsComponent instances matching criteria

```
List<T> loadAll(NutsSupportLevelContext<B> criteria)
```

criteria to match

⚙ loadBest(criteria)

load best NutsComponent instance matching criteria

```
T loadBest(NutsSupportLevelContext<B> criteria)
```

criteria to match

☕ NutsSessionTerminalBase

```
public net.vpc.app.nuts.NutsSessionTerminalBase
```

Session Terminal Base instance are special Terminal Base classes instances that handle workspace session.

☕ NutsSupportLevelContext

```
public net.vpc.app.nuts.NutsSupportLevelContext
```

⌚ Instance Properties

📋⌚ constraints

```
[read-only] T constraints  
T getConstraints()
```

📋⌚ workspace

```
[read-only] NutsWorkspace workspace  
NutsWorkspace getWorkspace()
```

☕ NutsSystemTerminalBase

```
public net.vpc.app.nuts.NutsSystemTerminalBase
```

Created by vpc on 2/20/17.

☕ NutsTerminalBase

```
public net.vpc.app.nuts.NutsTerminalBase
```

Created by vpc on 2/20/17.

⌚ Instance Properties

 **err**

```
[read-only] PrintStream err  
PrintStream getErr()
```

 **errMode**

```
[read-write] NutsTerminalBase errMode  
NutsTerminalMode getErrMode()  
NutsTerminalBase setErrMode(mode)
```

 **in**

```
[read-only] InputStream in  
InputStream getIn()
```

 **out**

```
[read-only] PrintStream out  
PrintStream getOut()
```

 **outMode**

```
[read-write] NutsTerminalBase outMode  
NutsTerminalMode getOutMode()  
NutsTerminalBase setOutMode(mode)
```

 **parent**

```
[read-only] NutsTerminalBase parent  
NutsTerminalBase getParent()
```

⚙ Instance Methods

⚙ `readLine(out, prompt, params)`

```
String readLine(PrintStream out, String prompt, Object params)
```

null null null

⚙ `readPassword(out, prompt, params)`

```
char[] readPassword(PrintStream out, String prompt, Object params)
```

null null null

☕ NutsTransportComponent

```
public net.vpc.app.nuts.NutsTransportComponent
```

Transport component responsible of creating a connexion to remote servers. Should handle at least valid http connections.

⚙ Instance Methods

⚙ `open(url)`

open url and return a valid `NutsTransportConnection`

```
NutsTransportConnection open(String url)
```

url to open

☕ NutsTransportConnection

```
public net.vpc.app.nuts.NutsTransportConnection
```

Connection to a remote server.

[Instance Properties](#)

uRLHeader

parse connection header and return meaningful information

```
[read-only] NutsURLHeader uRLHeader  
NutsURLHeader getURLHeader\(\)
```

Instance Methods

[open\(\)](#)

option connection and retrieve input stream

```
InputStream open\(\)
```

[upload\(parts\)](#)

parse connection header and return meaningful information

```
InputStream upload\(NutsTransportParamPart parts\)
```

parts to upload

[NutsTransportParamBinaryFilePart](#)

```
public net.vpc.app.nuts.NutsTransportParamBinaryFilePart
```

Created by vpc on 1/8/17.

[Constructors](#)

🔗 NutsTransportParamBinaryFilePart(name, fileName, value)

```
NutsTransportParamBinaryFilePart(String name, String fileName, Path value)
```

null null null

📄 Instance Properties

📄 📄 fileName

```
[read-only] public String fileName  
private final String fileName  
public String getFileName()
```

📄 📄 name

```
[read-only] public String name  
private final String name  
public String getName()
```

📄 📄 value

```
[read-only] public Path value  
private final Path value  
public Path getValue()
```

⚙️ Instance Methods

⚙️ equals(o)

```
boolean equals(Object o)
```

null

 hashCode()

```
int hashCode()
```

 toString()

```
String toString()
```

 NutsTransportParamBinaryStreamPart

```
public net.vpc.app.nuts.NutsTransportParamBinaryStreamPart
```

Created by vpc on 1/8/17.

 Constructors

 NutsTransportParamBinaryStreamPart(name, fileName, value)

```
NutsTransportParamBinaryStreamPart(String name, String fileName, InputStream  
value)
```

null null null

 Instance Properties

  fileName

```
[read-only] public String fileName  
private final String fileName  
public String getFileName()
```

  name

```
[read-only] public String name  
private final String name  
public String getName()
```

[value](#)

```
[read-only] public InputStream value  
private final InputStream value  
public InputStream getValue()
```

[NutsTransportParamParamPart](#)

```
public net.vpc.app.nuts.NutsTransportParamParamPart
```

Created by vpc on 1/8/17.

[Constructors](#)

[NutsTransportParamParamPart\(name, value\)](#)

```
NutsTransportParamParamPart(String name, String value)
```

null null

[Instance Properties](#)

[name](#)

```
[read-only] public String name  
private final String name  
public String getName()
```

[value](#)

```
[read-only] public String value  
private final String value  
public String getValue()
```

⚙ Instance Methods

⚙ equals(o)

```
boolean equals(Object o)
```

null

⚙ hashCode()

```
int hashCode()
```

⚙ toString()

```
String toString()
```

📄 NutsTransportParamPart

```
public net.vpc.app.nuts.NutsTransportParamPart
```

Created by vpc on 1/8/17.

📄 NutsTransportParamTextFilePart

```
public net.vpc.app.nuts.NutsTransportParamTextFilePart
```

Created by vpc on 1/8/17.

Constructors

NutsTransportParamTextFilePart(name, fileName, value)

```
NutsTransportParamTextFilePart(String name, String fileName, Path value)
```

null null null

Instance Properties

fileName

```
[read-only] public String fileName  
private final String fileName  
public String getFileName()
```

name

```
[read-only] public String name  
private final String name  
public String getName()
```

value

```
[read-only] public Path value  
private final Path value  
public Path getValue()
```

Instance Methods

equals(o)

```
boolean equals(Object o)
```

null

⚙️ hashCode()

```
int hashCode()
```

⚙️ toString()

```
String toString()
```

☕ NutsTransportParamTextReaderPart

```
public net.vpc.app.nuts.NutsTransportParamTextReaderPart
```

Created by vpc on 1/8/17.

∅ Constructors

∅ NutsTransportParamTextReaderPart(name, fileName, value)

```
NutsTransportParamTextReaderPart(String name, String fileName, Reader value)
```

null null null

∅ Instance Properties

📄∅ fileName

```
[read-only] public String fileName
private final String fileName
public String getFileName()
```

📄∅ name

```
[read-only] public String name  
private final String name  
public String getName()
```

📄 [value](#)

```
[read-only] public Reader value  
private final Reader value  
public Reader getValue()
```

⚙️ Instance Methods

⚙️ equals(o)

```
boolean equals(Object o)
```

null

⚙️ hashCode()

```
int hashCode()
```

⚙️ toString()

```
String toString()
```

☕ NutsURLHeader

```
public net.vpc.app.nuts.NutsURLHeader
```

url header meaning ful information

Instance Properties

contentEncoding

url content encoding

```
[read-only] String contentEncoding  
String getContentEncoding()
```

contentLength

url content length (file size)

```
[read-only] long contentLength  
long getContentLength()
```

contentType

url content type (file type)

```
[read-only] String contentType  
String getContentType()
```

lastModified

url content last modified

```
[read-only] Instant lastModified  
Instant getLastModified()
```

url

url value

```
[read-only] String url  
String getUrl()
```

☕ NutsUpdateRepositoryStatisticsCommand

```
public net.vpc.app.nuts.NutsUpdateRepositoryStatisticsCommand
```

Instance Properties

session

```
[write-only] NutsUpdateRepositoryStatisticsCommand session  
NutsUpdateRepositoryStatisticsCommand setSession(session)
```

Instance Methods

configure(skipUnsupported, args)

configure the current command with the given arguments. This is an override of the `NutsConfigurable#configure(boolean, java.lang.String...)` to help return a more specific return type;

```
NutsUpdateRepositoryStatisticsCommand configure(boolean skipUnsupported, String  
args)
```

when true, all unsupported options are skipped argument to configure with

run()

run this command and return `this` instance

```
NutsUpdateRepositoryStatisticsCommand run()
```

☕ NutsWorkspaceArchetypeComponent

```
public net.vpc.app.nuts.NutsWorkspaceArchetypeComponent
```

Created by vpc on 1/23/17.

Instance Properties

name

```
[read-only] String name  
String getName()
```

Instance Methods

initialize(session)

```
void initialize(NutsSession session)
```

null

NutsWorkspaceAware

```
public net.vpc.app.nuts.NutsWorkspaceAware
```

classes that implement this class will have their method `#setWorkspace(NutsWorkspace)` called upon its creation (by factory) with a non `null` argument to initialize. They may accept a call with a `null` argument later to dispose the instance.

====

Instance Properties

workspace

initialize or dispose the instance. when workspace is not null, the instance should initialize it values accordingly. when workspace is null, the instance should dispose resources.

```
[write-only] void workspace  
void setWorkspace(workspace)
```

NutsWorkspaceInitInformation

```
public net.vpc.app.nuts.NutsWorkspaceInitInformation
```

workspace initialization options.

Created by vpc on 1/23/17.

Instance Properties

apild

```
[read-only] String apiId  
String getApiId()
```

apiVersion

```
[read-only] String apiVersion  
String getApiVersion()
```

bootRepositories

```
[read-only] String bootRepositories  
String getBootRepositories()
```

bootWorkspaceFactory

```
[read-only] NutsBootWorkspaceFactory bootWorkspaceFactory  
NutsBootWorkspaceFactory getBootWorkspaceFactory()
```

classWorldLoader

```
[read-only] ClassLoader classWorldLoader  
ClassLoader getClassWorldLoader()
```

classWorldURLs

```
[read-only] URL[] classWorldURLs  
URL[] getClassWorldURLs()
```

extensionDependencies

```
[read-only] String extensionDependencies  
String getExtensionDependencies()
```

extensionDependenciesSet

```
[read-only] Set<String> extensionDependenciesSet  
Set<String> getExtensionDependenciesSet()
```

javaCommand

```
[read-only] String javaCommand  
String getJavaCommand()
```

javaOptions

```
[read-only] String javaOptions  
String getJavaOptions()
```

name

```
[read-only] String name  
String getName()
```

options

```
[read-only] NutsWorkspaceOptions options  
NutsWorkspaceOptions getOptions()
```

📄 📄 repositoryStoreLocationStrategy

```
[read-only] NutsStoreLocationStrategy repositoryStoreLocationStrategy  
NutsStoreLocationStrategy getRepositoryStoreLocationStrategy()
```

📄 📄 runtimeDependencies

```
[read-only] String runtimeDependencies  
String getRuntimeDependencies()
```

📄 📄 runtimeDependenciesSet

```
[read-only] Set<String> runtimeDependenciesSet  
Set<String> getRuntimeDependenciesSet()
```

📄 📄 runtimeId

```
[read-only] String runtimeId  
String getRuntimeId()
```

📄 📄 storeLocationLayout

```
[read-only] NutsOsFamily storeLocationLayout  
NutsOsFamily getStoreLocationLayout()
```

📄 📄 storeLocationStrategy

```
[read-only] NutsStoreLocationStrategy storeLocationStrategy  
NutsStoreLocationStrategy getStoreLocationStrategy()
```

📄 📄 uuid

```
[read-only] String uuid  
String getUuid()
```

📄 workspaceLocation

```
[read-only] String workspaceLocation  
String getWorkspaceLocation()
```

⚙️ Instance Methods

⚙️ getStoreLocation(location)

```
String getStoreLocation(NutsStoreLocation location)
```

null