dnf-daemon Documentation

Release 0.1.4

Tim Lauridsen

October 10, 2014

Contents

Contents:

CHAPTER 1

DBus service API documentation

The dnfdaemon is an easy way to utililize the power of the dnf package manager from your own programs

1.1 Data structures

Table 1.1: Data structures

Name	Content
Package Id (pkg_id)	“name,epoch,ver,rel,arch,repo_id”

Table 1.2: Attribute Descriptions

Attribute	Description
name	Package Name
epoch	Package Epoch
ver	Package Version
rel	Package Release
arch	Package Architecture
repo_id	Repository Id

Transaction result:

```
<transaction_result> ::= <result>, <result>, ...., <result>
<result>             ::= <action>, <pkg_list>
<action>             ::= install | update | remove | reinstall | downgrade | obsolete
<plg_list>           ::= <pkg_info>, <pkg_info>, ......, <pkg_info>
<pkg_info>           ::= <pkg_id>, size, <obs_list>
<pkg_id>             ::= name, epoch, version, release, arch, repo_id
<obs_list>           ::= <obs_id>, <obs_id>, ...., <obs_id>
<obs_id>             ::= name, epoch, version, release, arch, repo_id for packages obsoletes by <pkg_
```

CHAPTER 2

System Service

Table 2.1: DBus Names

Attribute	Value
object	org.baseurl.DnfSystem
interface	org.baseurl.DnfSystem
path	/

2.1 Misc methods

GetVersion()
Get the API version

Returns string with API version

Lock()
Get the daemon Lock, if posible

Unlock()
Release the daemon Lock, if posible

SetWatchdogState(*state*)
Set the Watchdog state. if watchdog is enabled, then daemon will exit not used in 20s.

Parameters **state** (*boolean (b)*) – True = Watchdog active, False = Watchdog disabled

2.2 Repository and config methods

GetRepositories(*filter*)
Get the value a list of repo ids

Parameters **filter** (*string*) – filter to limit the listed repositories

Returns list of repo id's

Return type array for stings (as)

GetRepo(*repo_id*)
Get information about a give repo_id

Parameters **repo_id** (*string*) – repo id

Returns a dictionary with repo information **(JSON)**

Return type string (s)

SetEnabledRepos(repo_ids):
Enabled a list of repositories, disabled all other repos

Parameters **repo_ids** – list of repo ids to enable

GetConfig(*setting*)
Get the value of a yum config setting

Parameters **setting** (*string*) – name of setting (debuglevel etc..)

Returns the config value of the requested setting **(JSON)**

Return type string (s)

SetConfig(*setting*, *value*)
Get the value of a yum config setting

Parameters

- **setting** (*string*) – name of setting (debuglevel etc..)
- **value** (misc types **(JSON)**) – name of setting (debuglevel etc..)

Returns did the update succed

Return type boolean (b)

ExpireCache()
Expire the dnf cache, to force dnf to check for updated metadata.

2.3 Package methods

These methods is for getting packages and information about packages

GetPackages(*pkg_filter*, *fields*)

Get a list of pkg list for a given package filter
each pkg list contains [pkg_id, field,....] where field is a atrribute of the package object
Ex. summary, size etc.
if no extra fields values are needed just use a empty array []

Parameters

- **pkg_filter** (*string*) – package filter ('installed','available','updates','obsoletes','recent','extras')
- **fields** (*array of strings (as)*) – yum package objects attributes to get.

Returns list of (id, field1, field2...) **(JSON)**, each JSON Sting contains (id, field1, field2...)

Return type array of strings (as)

GetPackagesByName(*name*, *attrs*, *newest_only*)
Get a list of pkg ids for starts with name and some user defined attributes

Parameters

- **name** (*string*) – name prefix to match

- **attrs** (*list of strings*) – a list of packages attributes to return
- **newest_only** (*boolean*) – show only the newest match or every match.

Returns list of pkg_id or [pkg_id, attr1, attr2, ...] if attrs is set

Return type array of strings (as)

GetAttribute (*id*, *attr*)
get yum package attribute (description, filelist, changelog etc)

Parameters

- **pkg_id** (*string*) – pkg_id to get attribute from
- **attr** (*string*) – name of attribute to get

Returns the value of the attribute **(JSON)**, the content depend on attribute being read

Return type string (s)

Search (*fields*, *keys*, *attrs*, *match_all*, *newest_only*, *tags*)
Search for packages where keys is matched in fields and return extra attributes

Parameters

- **fields** (*array of strings*) – yum po attributes to search in
- **keys** (*array of strings*) – keys to search for
- **attrs** – list of extra package attributes to get
- **match_all** (*boolean*) – match all keys or only one
- **newest_only** (*boolean*) – match all keys or only one
- **tags** (*boolean*) – search in pkgtags

Returns list of [pkg_id, attr1, attr2, ..] **JSON**

Return type string (s)

2.4 High level methods

The high level methods simulate basic dnf command line main functions.

Install (*cmds*)

Works just like the `dnf install <cmds>` command line

param cmds package arguments separated by spaces

type cmds string

return return code, result of resolved transaction (rc = 1 is ok, else failure) **(JSON)**

rtype string (s)

Remove (*cmds*)
Works just like the `dnf remove <cmds>` command line

Parameters **cmds** (*string*) – package arguments separated by spaces

Returns return code, result of resolved transaction (rc = 1 is ok, else failure) **(JSON)**

Return type string (s)

Update(*cmds*)
Works just like the `dnf update <cmds>` command line

Parameters **cmds** (*string*) – package arguments separated by spaces

Returns return code, result of resolved transaction (rc = 1 is ok, else failure) **(JSON)**

Return type string (s)

Reinstall(*cmds*)
Works just like the `dnf reinstall <cmds>` command line

Parameters **cmds** (*string*) – package arguments separated by spaces

Returns return code, result of resolved transaction (rc = 1 is ok, else failure) **(JSON)**

Return type string (s)

Downgrade(*cmds*)
Works just like the `dnf downgrade <cmds>` command line

Parameters **cmds** (*string*) – package arguments separated by spaces

Returns return code, result of resolved transaction (rc = 1 is ok, else failure) **(JSON)**

Return type string (s)

2.5 Transaction methods

These methods is for handling the current yum transaction

AddTransaction(*id*, *action*)
Add an package to the current transaction

Parameters

- **id** (*string*) – package id for the package to add
- **action** (*string*) – the action to perform (install, update, remove, obsolete, reinstall, downgrade, localinstall)

Returns list of (pkg_id, transaction state) pairs for the added members (comma separated)

Return type array of strings (as)

ClearTransaction()
Clear the current transaction

GetTransaction()
Get the currrent transaction

Returns list of (pkg_id, transaction state) pairs in the current transaction (comma separated)

Return type array of strings (as)

BuildTransaction()
Depsolve the current transaction

Returns (return code, result of resolved transaction) pair (rc = 1 is ok, else failure) **(JSON)**

Return type string (s)

RunTransaction(*max_err*)
Execute the current transaction

Parameters **max_err** (*integer (i)*) – maximum download errors before we bail out

Returns (rc,msg) rc = state of run transaction (0 = ok, 1 = need GPG import confirmation, 2 = error) and msgs = list of error messages **(JSON)**

Return type string (s)

ConfirmGPGImport (*self*, *hexkeyid*, *confirmed*)
Confirm import of at GPG Key by yum

Parameters

- **hexkeyid** (*string (s)*) – hex keyid for GPG key
- **confirmed** (*boolean (b)*) – confirm import of key (True/False)

2.6 Groups

Methods to work with yum groups and categories

GetGroups ()
Get available Categories & Groups

GetGroupPackages (*grp_id*, *grp_flt*)
Get packages in a group by grp_id and grp_flt

Parameters

- **grp_id** (*string (s)*) – The Group id
- **grp_flt** (*string (s)*) – Group Filter (all or default)

Returns list of pkg_id's

Return type array of strings (as)

GroupInstall (*patterns*)
Install groups matching patterns

Parameters **patterns** – patterns separated by ' ' (ex. "firefox xfce-desktop")

GroupRemove (*patterns*)
Removing groups matching patterns

Parameters **patterns** – patterns separated by ' ' (ex. "firefox xfce-desktop")

Note: Under Development

More to come in the future, methods to install groups etc. has to be defined and implemented

---

2.7 History

Methods to work with the package transaction history

GetHistoryByDays (*start_days*, *end_days*)
Get History transaction in a interval of days from today

Parameters

- **start_days** (*integer*) – start of interval in days from now (0 = today)

- **end_days** (*integer*) – end of interval in days from now

Returns a list of (transaction ids, date-time) pairs (JSON) :rtype: string (s)

GetHistoryPackages (*tid*)
Get packages from a given yum history transaction id

Parameters **tid** (*integer*) – history transaction id

Returns list of (pkg_id, state, installed) pairs

Return type json encoded string

HistorySearch (*pattern*)
Search the history for transaction matching a pattern

Parameters **pattern** (*string*) – patterne to match

Returns list of (tid,isodates)

2.8 Signals

TransactionEvent(self,event,data):
Signal with Transaction event information, telling the current step in the processing of the current transaction.

Steps are : start-run, download, pkg-to-download, signature-check, run-test-transaction, run-transaction, verify, fail, end-run

Parameters **event** – current step

RPMProgress(self, package, action, te_current, te_total, ts_current, ts_total):
signal with RPM Progress

Parameters

- **package** – A package object or simple string of a package name
- **action** – action current performed on the package: install, cleanup, remove etc.
- **te_current** – Current number of bytes processed in the transaction element being processed
- **te_total** – Total number of bytes in the transaction element being processed
- **ts_current** – number of processes completed in whole transaction
- **ts_total** – total number of processes in the transaction.

GPGImport(self, pkg_id, userid, hexkeyid, keyurl, timestamp):
signal with GPG Key information of a key there need to be confirmed to complete the current transaction. after signal is send transaction will abort with rc=1 Use ConfirmGPGImport method to comfirm the key and run RunTransaction again

Parameters

- **pkg_id** – pkg_id for the package needing the GPG Key to be verified
- **userid** – GPG key name
- **hexkeyid** – GPG key hex id
- **keyurl** – Url to the GPG Key
- **timestamp** – GPG Timestamp

ErrorMessage(*self*, *error_msg*)
An error message from the backend

Parameters **error_msg** – error message (s)

DownloadStart(*self*, *num_files*, *num_bytes*)
Starting a new parallel download batch

Parameters

- **num_files** – number of files to download
- **num_bytes** – number of bytes to download

DownloadProgress(*self*, *name*, *frac*, *total_frac*, *total_files*)
Progress for a single instance in the batch

Parameters

- **name** – name of package
- **frac** – fraction downloaded (0.0 -> 1.0)
- **total_frac** – fraction downloaded of whole batch(0.0 -> 1.0)
- **total_files** – total files downloaded

DownloadEnd(*self*, *name*, *status*, *msg*)
Download of af single instace ended

Parameters

- **name** – name of package
- **status** – download status
- **msg** – error message, if status != ok

RepoMetaDataProgress(*self*, *name*, *frac*)
Repository Metadata Download progress

Parameters

- **name** – repository id
- **frac** – fraction downloaded (0.0 -> 1.0)

CHAPTER 3

Session Service

Table 3.1: DBus Names

Attribute	Value
object	org.baseurl.DnfSession
interface	org.baseurl.DnfSession
path	/

3.1 Misc methods

GetVersion()
Get the API version

Returns string with API version

Lock()
Get the daemon Lock, if posible

Unlock()
Release the daemon Lock, if posible

3.2 Repository and config methods

GetRepositories(*filter*)
Get the value a list of repo ids

Parameters **filter** (*string*) – filter to limit the listed repositories

Returns list of repo id's

Return type array for stings (as)

GetRepo(*repo_id*)
Get information about a give repo_id

Parameters **repo_id** (*string*) – repo id

Returns a dictionary with repo information **(JSON)**

Return type string (s)

SetEnabledRepos(repo_ids):
Enabled a list of repositories, disabled all other repos

Parameters **repo_ids** – list of repo ids to enable

GetConfig(*setting*)
Get the value of a yum config setting

Parameters **setting** (*string*) – name of setting (debuglevel etc..)

Returns the config value of the requested setting **(JSON)**

Return type string (s)

3.3 Package methods

These methods is for getting packages and information about packages

GetPackages(*pkg_filter*)
get a list of packages matching the filter type

Parameters **pkg_filter** (*string*) – package filter ('installed','available','updates','obsoletes','recent','extras')

Returns list of pkg_id's

Return type array of strings (as)

GetPackageWithAttributes(*pkg_filter*, *fields*)

Get a list of pkg list for a given package filter
each pkg list contains [pkg_id, field,....] where field is a atrribute of the package object
Ex. summary, size etc.

Parameters

- **pkg_filter** (*string*) – package filter ('installed','available','updates','obsoletes','recent','extras')
- **fields** (*array of strings (as)*) – yum package objects attributes to get.

Returns list of (id, field1, field2...) **(JSON)**, each JSON Sting contains (id, field1, field2...)

Return type array of strings (as)

GetPackagesByName(*name*, *attrs*, *newest_only*)
Get a list of pkg ids for starts with name and some user defined attributes

Parameters

- **name** (*string*) – name prefix to match
- **attrs** (*list of strings*) – a list of packages attributes to return
- **newest_only** (*boolean*) – show only the newest match or every match.

Returns list of pkg_id or [pkg_id, attr1, attr2, ...] if attrs is set

Return type array of strings (as)

GetAttribute(*id*, *attr*)
get yum package attribute (description, filelist, changelog etc)

Parameters

- **pkg_id** (*string*) – pkg_id to get attribute from
- **attr** (*string*) – name of attribute to get

Returns the value of the attribute **(JSON)**, the content depend on attribute being read

Return type string (s)

Search (*fields*, *keys*, *attrs*, *match_all*, *newest_only*, *tags*)
Search for packages where keys is matched in fields and return extra attributes

Parameters

- **fields** (*array of strings*) – yum po attributes to search in
- **keys** (*array of strings*) – keys to search for
- **attrs** – list of extra package attributes to get
- **match_all** (*boolean*) – match all keys or only one
- **newest_only** (*boolean*) – match all keys or only one
- **tags** (*boolean*) – search in pkgtags

Returns list of pkg_id or [pkg_id, attr1, attr2, ..] if attr is defined **JSON**

Return type string (s)

3.4 Groups

Methods to work with dnf groups and categories

GetGroups ()
Get available Categories & Groups

GetGroupPackages (*grp_id*, *grp_flt*)
Get packages in a group by grp_id and grp_flt

Parameters

- **grp_id** (*string (s)*) – The Group id
- **grp_flt** (*string (s)*) – Group Filter (all or default)

Returns list of pkg_id's

Return type array of strings (as)

Note: Under Development

More to come in the future, methods to install groups etc. has to be defined and implemented

---

3.5 Signals

Note: Under Development

Signals is not documented yet

CHAPTER 4

Client API for Python 2.x and 3.x

4.1 Classes

4.1.1 System API

4.1.2 Session API

4.2 Exceptions

class `DaemonError` (*Exception*)

Base Exception from the backend

class `AccessDeniedError` (*DaemonError*)

PolicyKit access was denied.

Ex. User press cancel button in policykit window

class `LockedError` (*DaemonError*)

dnf is locked by another application

Ex. dnf is running in a another session You have not called the Lock method to grep the Lock

class `TransactionError` (*DaemonError*)

Error in the dnf transaction.

CHAPTER 5

Examples

5.1 Python 2.x &3.x

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*