THE MITRE CORPORATION

The OVAL® Language UNIX Component Model Specification

Version 5.11

Danny Haynes, Stelios Melachrinoudis

12/18/2014

The Open Vulnerability and Assessment Language (OVAL®) is an international, information security, community standard to promote open and publicly available security content, and to standardize the transfer of this information across the entire spectrum of security tools and services. By standardizing the three main steps of the assessment process: representing configuration information of systems for testing; analyzing the system for the presence of the specified machine state; and reporting the results of the assessment, the OVAL Language provides a common and structured format that facilitates collaboration and information sharing among the information security community as well as interoperability among tools. This document defines the UNIX platform-specific data model for the OVAL Language.

Acknowledgements

Trademark Information

Warnings

Feedback

The MITRE Corporation welcomes any feedback regarding the OVAL Language UNIX Component Model Specification. Please send any comments, questions, or suggestions to the public OVAL Developer's Forum at oval-developer-list@lists.mitre.org or directly to the OVAL Moderator at oval@mitre.org[2].

[1] For more information see https://oval.mitre.org/about/termsofuse.html

[2] For more information see https://oval.mitre.org/

Contents

1. Introduction

1.1 Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in *RFC 2119* [1].

The following font and font style conventions are used throughout the remainder of this document:

- The `Courier New` font without formatting is used for writing constructs in the OVAL Language Data Model. When the font is **`boldfaced`**, it indicates commands on the UNIX command line.
  Examples: `generator` (OVAL Construct), **`ls -al`** (UNIX command)
- The *'italic, with single quotes'* font is used for noting values for OVAL Language properties.
  Example: *'does not exist'*
- The bold font and the keyword **Default Value:** are used to indicate a property's default value.
  Example: **Default Value: -1**
- The bold font and the keyword **xsi:nil="true":** are used to indicate the meaning of an entity when the xsi:nil property is set to true.
  Example: **xsi:nil="true"** indicates that the `file_object` MUST collect the set of directories specified by the path entity. In addition, a value, for the filename entity, MUST NOT be specified.

This document uses the concept of namespaces[3] to logically group OVAL constructs throughout both the Data Model section of the document, as well as other parts of the specification. The format of these namespaces is `prefix:element`, where the prefix is the namespace component, and the element is the name of the qualified construct. The following table lists the namespaces used in this document:

Data Model	Namespace	Description	Example
OVAL Definitions	oval-def	The OVAL Definitions data model that defines the core framework constructs for creating OVAL Definitions. This is defined in the OVAL Language Specification [2].	`oval-def:TestType`
OVAL System Characteristics	oval-sc	The OVAL System Characteristics data model, which defines the constructs used to capture the data collected on a target system. This is defined in the OVAL Language Specification.	`oval-sc:ItemType`
UNIX Definitions	unix-def	The UNIX Definitions data model defines the platform-specific	`unix-def:file_test`

[3] For more information see http://en.wikipedia.org/wiki/Namespace_(computer_science)

		constructs used in OVAL Definitions to make assertions about the state of UNIX systems.	
UNIX System Characteristics	unix-sc	The UNIX System Characteristics data model defines the platform-specific constructs used in OVAL System Characteristics to represent the system state information collected from UNIX systems.	`unix-sc:file_item`

Lastly, each OVAL Test will contain a section titled "Known Supported Platforms" that specifies which platforms the OVAL Test is known to work on. This section is provided for convenience only and should not be considered a comprehensive list. In addition, there may be further known support restrictions specified for behaviors or entities that supersede the "Known Supported Platforms" section for the OVAL Test.

1.2 Document Structure

This document serves as the specification for the UNIX extension of the OVAL Language Specification and defines the platform-specific data model. This document is organized into the following sections:

- Section 1 – Introduction
- Section 2 – OVAL Language UNIX Component Model
- Appendix A – References
- Appendix B – Change Log
- Appendix C – Terms and Acronyms

2. OVAL Language UNIX Component Model

The OVAL Language UNIX Component Data Model is the platform-specific extension of the OVAL Language Data Model for UNIX operating systems.

2.1 Data Model Conventions

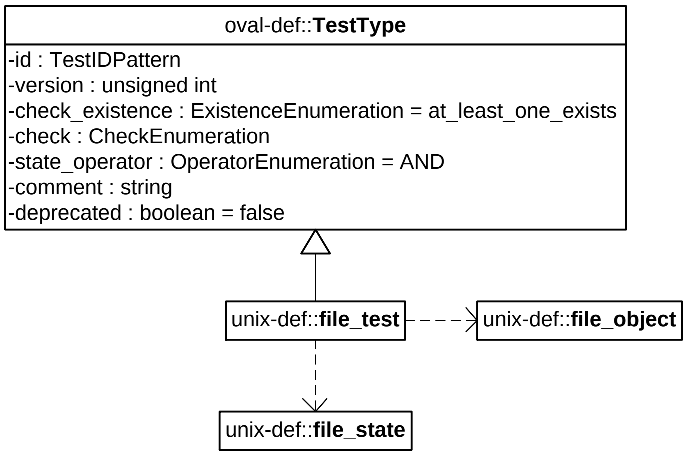
This document follows the data model conventions described in Section 4.1 of the OVAL Language Specification.

2.2 unix-def:file_test

The `file_test` is used to make assertions about the metadata associated with the directories and files returned by either an **`ls`**[4] command, **`stat`**[5] command, or **`stat()`**[6] system call, on file systems

[4] For more information see http://linux.die.net/man/1/ls
[5] For more information see http://linux.die.net/man/1/stat
[6] For more information see http://linux.die.net/man/2/stat

supported by UNIX operating systems. The `file_test` MUST reference one `file_object` and zero or more `file_states`.

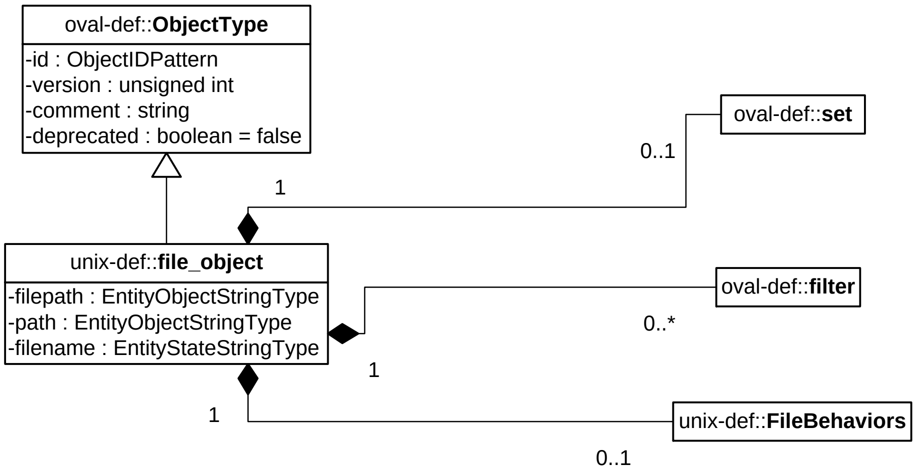

oval-def::TestType
-id : TestIDPattern
-version : unsigned int
-check_existence : ExistenceEnumeration = at_least_one_exists
-check : CheckEnumeration
-state_operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::file_test
unix-def::file_object
unix-def::file_state

2.2.1 Known Supported Platforms

- Red Hat Enterprise Linux 5
- Mac OSX 10.6
- Solaris 10

2.3 unix-def:file_object

The `file_object` construct defines the set of files and/or directories whose associated system state information should be collected and represented as `file_items`. The `file_object` is capable of collecting all UNIX file types (directory, regular file, character device, block device, fifo, symbolic link, and socket). The set of files to be evaluated may be identified with either a complete filepath or a path and filename. Only one of these options may be selected.



Property	Type	Multiplicity	Nillable	Description
set	oval-def:set	0..1	false	Enables the expression of complex `file_objects` that are the result of logically combining and filtering the `file_items` that are identified by one or more `file_objects`. The behaviors, filepath, path, filename, and filter properties MUST NOT be specified when this property is specified. Please see the OVAL Language Specification for additional information.
behaviors	unix-def:FileBehaviors	0..1	false	Specifies the behaviors that direct how the `file_object` collects `file_items` from the system.
filepath	oval-def: EntityObjectStringType	0..1	false	The absolute path to a file on the system. A directory MUST NOT be specified for this property, and the path and filename properties MUST NOT be specified when this property is specified. The max_depth, recurse, and recurse_direction behaviors MUST NOT be used in conjunction with this property as they are reserved for use with the path and filename properties. This is because the filepath property represents an absolute path to a particular file and it is not possible to recurse over a file. Also, the `recurse_file_system` behavior MUST NOT be set to 'defined' when a pattern match is used with a filepath property.
path	oval-def: EntityObjectStringType	0..1	false	The directory component of the absolute path to a directory or file on the system. The filepath property MUST NOT be specified when this property is specified. When a pattern match is used with a path entity, the max_depth, recurse_direction, and recurse behaviors MUST NOT be used.

				Also, the recurse_file_system behavior MUST NOT be set to 'defined' when a pattern match is used with a path property.
filename	oval-def: EntityObjectStringType	0..1	true	The name of a file to evaluate. A filename SHOULD NOT contain the NUL or / characters[7]. In addition, a filename SHOULD NOT 1) include control characters and shell metacharacters such as those in the set {*, ?, :, [,], ", <, >, |, (,), {, }, &, ', !, \, ;} or 2) start with a dash (-)[8], due to the potentially dangerous consequences associated with the unintended use of certain UNIX commands. The filepath property MUST NOT be specified when this property is specified. **xsi:nil="true"** indicates that the `file_object` MUST collect the set of directories specified by the path entity. In addition, a value for the filename entity MUST NOT be specified or a var_ref is used.
filter	oval-def:filter	0..*	false	Allows for the explicit inclusion or exclusion of `file_items` from the set of `file_items` collected by a `file_object`. Please see the OVAL Language Specification [2] for additional information.

2.4 unix-def:FileBehaviors

The `FileBehaviors` construct defines the behaviors that direct how the `file_object` collects `file_items` from the system. Note that using these behaviors may result in some unique results. For example, a double negative type condition might be created where an object entity says include everything except a specific item, but a behavior is used that might then add that item back in.

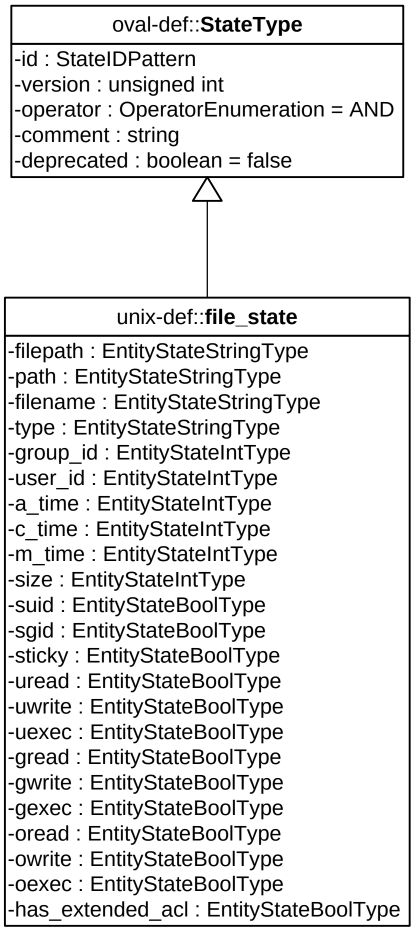
[7] For more information see http://www.dwheeler.com/essays/fixing-unix-linux-filenames.html
[8] For more information see http://www.dwheeler.com/essays/fixing-unix-linux-filenames.html#metacharacters

Attribute	Type	Possible Values	Description
max_depth	integer	*< -1* *-1* *0* *> 0*	Defines the maximum depth of file system traversal when the recurse_direction behavior is set to a value other than *'none'*. < *-1*: not permitted. *-1***:** traverse the file system with no limitation. *0***:** do not traverse the file system. > *0***:** traverse the file system for the specified number of levels. **Default Value: -1**
recurse	string	*'none'* *'files'* *'files and directories'* *'symlinks'* *'directories'* *'symlinks and directories'*	Defines how to recurse into the path entity, i.e. what to follow during recursion. Options include symlinks, directories, or both. A max-depth other than 0 MUST be specified for recursion to take place. *'none'*: **DEPRECATED (5.4)** None was originally intended to mean no recusion; however, this is already covered by the `recurse_direction` attribute, and so it has been deprecated with removal in version 6.0. *'files'*: **DEPRECATED (5.4)** This value has been deprecated in 5.4 and will be removed in version 6.0 because it is not possible to recurse files. *'files and directories'*: **DEPRECATED (5.4)** This value has been deprecated in 5.4 and will be removed in version 6.0 because it is not possible to recurse files. *'symlinks'*: Traverse via only symlinks. *'directories'*: Traverse via only directories. *'symlinks and directories'*: Traverse via both symlinks and directories.

recurse_direction	string	*'none'* 'up' *'down'*	Defines the direction to recursively visit the directories on the file system. *'none'*: do not traverse the file system. 'up': traverse the file system by recursively visiting the parent directories. *'down'*: traverse the file system by recursively visiting the child directories. An error MUST NOT be reported when the max_depth behavior specifies a certain level of traversal and that level does not exist. **Default Value: none**
recurse_file_system	string	*'all'* *'local'* *'defined'*	Defines the file system limitation of any searching. This applies to all operations as specified in the path or filepath entity. In most cases it is recommended that the value of *'local'* be used to ensure that file system searching is limited to only the local file systems, as searching 'all' file systems may have performance implications. *'all'*: traverse both local and remote file systems. *'local'*: only traverse the local file systems. *'defined'*: only traverse the specified file system. The value of *'defined'* MUST only be used in conjunction with the equality operation because the path or filepath entity must explicitly define a file system. **Default Value: all**

2.5 unix-def:file_state

The `file_state` construct is used by a `file_test` to specify the system state information, associated with files or directories, to check on file systems that are supported by UNIX platforms. All of the parameters here can be found via the `stat` command[9] and system call on a per file basis, or for all files and directories, **`ls -al`**, **`ls -alu`**, or **`ls -alc`** where appropriate[10] (except for the group and user numbers). For convenience in identifying permissions, the user that each permission refers to is underlined and boldfaced (owner/user, group, or other) as part of the ten character string outputted from the command **`ls -l`**, `drwxrwxrwx`. For example, the d in **`d`** `rwx rwx rwx` represents a directory. For the s and t bits, capitalized letters (S and T) indicate that the execute permission is OFF, whereas lowercase letters indicate that the execute permission is ON[11].


oval-def::StateType
-id : StateIDPattern
-version : unsigned int
-operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::file_state
-filepath : EntityStateStringType
-path : EntityStateStringType
-filename : EntityStateStringType
-type : EntityStateStringType
-group_id : EntityStateIntType
-user_id : EntityStateIntType
-a_time : EntityStateIntType
-c_time : EntityStateIntType
-m_time : EntityStateIntType
-size : EntityStateIntType
-suid : EntityStateBoolType
-sgid : EntityStateBoolType
-sticky : EntityStateBoolType
-uread : EntityStateBoolType
-uwrite : EntityStateBoolType
-uexec : EntityStateBoolType
-gread : EntityStateBoolType
-gwrite : EntityStateBoolType
-gexec : EntityStateBoolType
-oread : EntityStateBoolType
-owrite : EntityStateBoolType
-oexec : EntityStateBoolType
-has_extended_acl : EntityStateBoolType

[9] For more information see http://linux.die.net/man/1/stat
[10] For more information see http://linux.die.net/man/1/ls
[11] For more information see http://evolt.org/node/263 and http://www.greenend.org.uk/rjk/tech/perms.html

Property	Type	Multiplicity	Nillable	Description
filepath	oval-def:EntityStateStringType	0..1	false	The absolute path to a file on the system. A directory MUST NOT be specified for this property. The max_depth and recurse_direction behaviors MUST NOT be used in conjunction with this property as they are reserved for use with the path and filename properties.
path	oval-def:EntityStateStringType	0..1	false	The directory component of the absolute path to a directory or file on the system.
filename	oval-def:EntityStateStringType	0..1	false	The name of a file to evaluate. A filename SHOULD NOT contain the NUL or / characters[12]. In addition, a filename SHOULD NOT 1) include control characters and shell metacharacters such as those in the set {*, ?, :, [,], ", <, >, |, (,), {, }, &, ', !, \, ;} or 2) start with a dash (-)[13], due to the potentially dangerous consequences associated with the unintended use of

[12] For more information see http://www.dwheeler.com/essays/fixing-unix-linux-filenames.html
[13] For more information see http://www.dwheeler.com/essays/fixing-unix-linux-filenames.html#metacharacters

				certain UNIX commands. The filepath property MUST NOT be specified when this property is specified.
type	oval-def:EntityStateStringType	0..1	false	The file's type: regular file (regular), directory, named pipe (fifo), symbolic link, socket or block special. In the output for the **`stat`** command, this information is found right after the IO Block field[14], and for the output of the **`ls -l`** command[15], **d** `rwx rwx rwx`.
group_id	oval-def:EntityStateIntType	0..1	false	The group owner of a file, by group number. This can be found via the **`stat`** command[16].
user_id	oval-def:EntityStateIntType	0..1	false	The numeric user id, or uid, is the third column of each user's entry in /etc/passwd. This element represents the owner of the file. This can be found via the **`stat`** command[17].
a_time	oval-def:EntityStateIntType	0..1	false	The time that the file was last accessed, in SECONDS, since the UNIX epoch, which is

[14] For more information see http://www.thegeekstuff.com/2009/07/unix-stat-command-how-to-identify-file-attributes/

[15] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html

[16] For more information see http://www.thegeekstuff.com/2009/07/unix-stat-command-how-to-identify-file-attributes/

[17] For more information see http://www.thegeekstuff.com/2009/07/unix-stat-command-how-to-identify-file-attributes/

				the time 00:00:00 UTC on January 1, 1970. Found via the **`ls –lu`** or **`stat`** commands.
c_time	oval-def:EntityStateIntType	0..1	false	The time that the file's inode was changed, in SECONDS, since the UNIX epoch, which is the time 00:00:00 UTC on January 1, 1970. Found via the **`ls –lc`**, or **`stat`** commands, or the **`stat`** system call.
m_time	oval-def:EntityStateIntType	0..1	false	The time, in seconds, that the file was last modified since the UNIX epoch, which is the time 00:00:00 UTC on January 1, 1970. Found via the **`ls –l`** or **`stat`** commands.
size	oval-def:EntityStateIntType	0..1	false	The size of the file in bytes. Both are indicated in the output of the **`ls –l`** and **`stat`** commands.
suid	oval-def:EntityStateBoolType	0..1	false	Indicates the program runs with the uid (thus privileges) of the file's owner, rather than the calling user. For the output of the **`ls –ld`** or **`stat`** command[18], it is indicated by `d rw`**<u>`S`</u>**` rwx rwx` where s replaces the first x.
sgid	oval-def:EntityStateBoolType	0..1	false	Indicates the program runs with the gid (thus privileges) of the file's group owner, rather than the calling user's

[18] For more information about the different types in the `ls –l` command see http://www.hackinglinuxexposed.com/articles/20030417.html

				group. For the output of the **`ls -ld`** or **`stat`** command[19] it is indicated by `d rwx rw`**<u>`s`</u>**` rwx` where s replaces the second x.
sticky	oval-def:EntityStateBoolType	0..1	false	Indicates that the users can delete each other's files in this directory, when said directory is writable by those users. For the output of the **`ls -ld`** or **`stat`** command[20] it is indicated by `d rwx rwx rw`**<u>`t`</u>** where t replaces the final x for a directory.
uread	oval-def:EntityStateBoolType	0..1	false	Indicates the owner (user owner) of the file can read this file, or if a directory, read the directory contents. For the output of the **`ls -l`** or **`stat`** command[21] it is indicated by `d `**<u>`r`</u>**`wx rwx rwx`.
uwrite	oval-def:EntityStateBoolType	0..1	false	Indicates the owner (user owner) of the file can write to this file, or if a directory, write to the directory. For the output of the **`ls -l`** or **`stat`** command[22] it is indicated by `d r`**<u>`w`</u>**`x rwx rwx`.

[19] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[20] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[21] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[22] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html

uexec	oval-def:EntityStateBoolType	0..1	false	Indicates the owner (user owner) of the file can execute it or, if a directory, change into the directory. For the output of the **`ls -l`** command[23] it is indicated by `d rw`**`x`** `rwx rwx`.
gread	oval-def:EntityStateBoolType	0..1	false	Indicates the group owner of the file can read this file, or if a directory, read the directory contents. For the output of the **`ls -l`** command[24] it is indicated by `d rwx` **`r`**`wx rwx`.
gwrite	oval-def:EntityStateBoolType	0..1	false	Indicates the group owner of the file can write to this file, or if a directory, write to the directory. For the output of the **`ls -l`** command[25] it is indicated by `d rwx r`**`w`**`x rwx`.
gexec	oval-def:EntityStateBoolType	0..1	false	Indicates the group owner of the file can execute it or, if a directory, change into the directory. For the output of the **`ls -l`** command[26] it is indicated by `d rwx rw`**`x`** `rwx`.
oread	oval-def:EntityStateBoolType	0..1	false	Indicates that all other users can read this

[23] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[24] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[25] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[26] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html

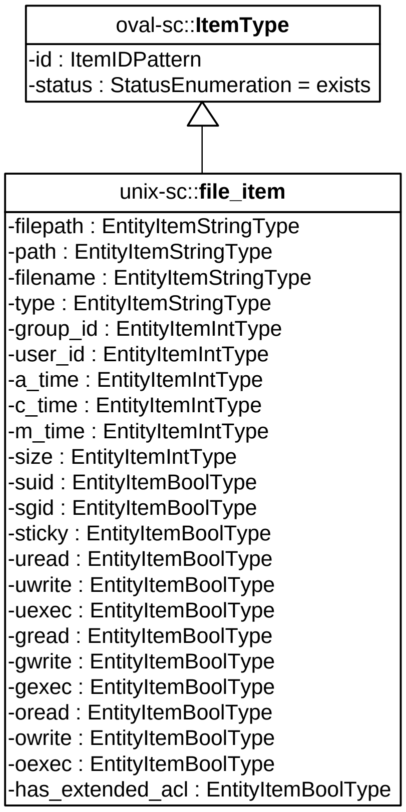
				file, or if a directory, read the directory contents. For the output of the **`ls -l`** command[27] it is indicated by `d rwx rwx` **`r`**`wx`.
owrite	oval-def:EntityStateBoolType	0..1	false	Indicates that all other users can write to this file, or if a directory, write to the directory. For the output of the **`ls -l`** command[28] it is indicated by `d rwx rwx r`**`w`**`x`.
oexec	oval-def:EntityStateBoolType	0..1	false	Indicates that all other users can execute the file or, if a directory, change into the directory. For the output of the **`ls -l`** command[29] it is indicated by `d rwx rwx rw`**`x`**.
has_extended_acl	oval-def:EntityStateBoolType	0..1	false	Indicates the file or directory has ACL permissions[30] applied to it. For the output of the **`ls -l`** or `stat` commands is it indicated by a plus sign (+) appended to the end of the `d rwx rwx rwx` string[31] as in `d rwx rwx rwx` **+**. If the file or directory doesn't have

[27] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[28] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[29] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[30] For more information see http://www.vanemery.com/Linux/ACL/linux-acl.html or http://www.softpanorama.info/Commercial_linuxes/linux_acl.shtml
[31] For more information see http://www.vanemery.com/Linux/ACL/linux-acl.html

				an ACL, or it matches the standard UNIX permissions, the value will be *false*. Otherwise if a file or directory has an ACL, the value will be *true*.

2.6 unix-sc:file_item

The `file_item` construct defines the system state information associated with files and directories on file systems supported by the UNIX platform. All of the parameters here can be found via the `stat` command[32] on a per file basis, or for all files and directories, **`ls -al`**, **`ls -alu`**, or **`ls -alc`** where appropriate[33] (except for the group and user numbers). For convenience in identifying permissions, the user that each permission refers to is underlined and boldfaced (owner/user, group, or other) as part of the ten character string outputted from the command **`ls -l`**, `drwxrwxrwx`. For example, the d in **`d`** `rwx rwx rwx` represents a directory. For the s and t bits, capitalized letters indicate that the execute permission is OFF, whereas lowercase letters indicate that the execute permission is ON[34].


oval-sc::ItemType
-id : ItemIDPattern
-status : StatusEnumeration = exists
unix-sc::file_item
-filepath : EntityItemStringType
-path : EntityItemStringType
-filename : EntityItemStringType
-type : EntityItemStringType
-group_id : EntityItemIntType
-user_id : EntityItemIntType
-a_time : EntityItemIntType
-c_time : EntityItemIntType
-m_time : EntityItemIntType
-size : EntityItemIntType
-suid : EntityItemBoolType
-sgid : EntityItemBoolType
-sticky : EntityItemBoolType
-uread : EntityItemBoolType
-uwrite : EntityItemBoolType
-uexec : EntityItemBoolType
-gread : EntityItemBoolType
-gwrite : EntityItemBoolType
-gexec : EntityItemBoolType
-oread : EntityItemBoolType
-owrite : EntityItemBoolType
-oexec : EntityItemBoolType
-has_extended_acl : EntityItemBoolType

[32] For more information see http://linux.die.net/man/1/stat
[33] For more information see http://linux.die.net/man/1/ls
[34] For more information see http://evolt.org/node/263

Property	Type	Multiplicity	Nillable	Description
filepath	oval-sc:EntityItemStringType	0..1	false	The absolute path to a file on the system. A directory MUST NOT be specified for this property. The max_depth and recurse_direction behaviors MUST NOT be used in conjunction with this property as they are reserved for use with the path and filename properties.
path	oval-sc:EntityItemStringType	0..1	false	The directory component of the absolute path to a directory or file on the system.
filename	oval-sc:EntityItemStringType	0..1	false	The name of a file to evaluate. A filename SHOULD NOT contain the NUL or / characters[35]. In addition, a filename SHOULD NOT 1) include control characters and shell metacharacters such as those in the set {*, ?, :, [,], ", <, >, |, (,), {, }, &, ', !, \, ;} or 2) start with a dash (-)[36], due to the potentially dangerous consequences associated with the unintended use of certain UNIX

[35] For more information see http://www.dwheeler.com/essays/fixing-unix-linux-filenames.html
[36] For more information see http://www.dwheeler.com/essays/fixing-unix-linux-filenames.html#metacharacters

				commands. The filepath property MUST NOT be specified when this property is specified.
type	oval-sc:EntityItemStringType	0..1	false	The file's type: regular file (regular), directory, named pipe (fifo), symbolic link, socket or block special. In the output for the **`stat`** command, this information is found right after the IO Block field[37], and for the output of the **`ls -l`** command[38], **d** `rwx rwx rwx`.
group_id	oval-sc:EntityItemIntType	0..1	false	The group owner of a file, by group number. This can be found via the **`stat`** command[39].
user_id	oval-sc:EntityItemIntType	0..1	false	The numeric user id, or uid, is the third column of each user's entry in /etc/passwd. This element represents the owner of the file. This can be found via the **`stat`** command[40].
a_time	oval-sc:EntityItemIntType	0..1	false	The time that the file was last accessed, in SECONDS, since the UNIX epoch, which is the time 00:00:00 UTC on January 1, 1970.

[37] For more information see http://www.thegeekstuff.com/2009/07/unix-stat-command-how-to-identify-file-attributes/
[38] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[39] For more information see http://www.thegeekstuff.com/2009/07/unix-stat-command-how-to-identify-file-attributes/
[40] For more information see http://www.thegeekstuff.com/2009/07/unix-stat-command-how-to-identify-file-attributes/

				Found via the **`ls -lu`** or **`stat`** commands.
c_time	oval-sc:EntityItemIntType	0..1	false	The time that the file's inode was changed, in SECONDS, since the UNIX epoch, which is the time 00:00:00 UTC on January 1, 1970. Found via the **`ls -lc`** or **`stat`** commands.
m_time	oval-sc:EntityItemIntType	0..1	false	The time, in seconds, that the file was last modified since the UNIX epoch, which is the time 00:00:00 UTC on January 1, 1970. Found via the **`ls -l`** or **`stat`** commands.
size	oval-sc:EntityItemIntType	0..1	false	The size of the file in bytes. Both are indicated in the output of the **`ls -l`** and **`stat`** commands.
suid	oval-sc:EntityItemBoolType	0..1	false	Indicates the program runs with the uid (thus privileges) of the file's owner, rather than the calling user. For the output of the **`ls -ld`** or **`stat`** command[41] it is indicated by `d rw`**s** `rwx rwx` where s replaces the first x.
sgid	oval-sc:EntityItemBoolType	0..1	false	Indicates the program runs with the gid (thus privileges) of the file's group owner, rather than the calling user's group. For the output

[41] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html

				of the **`ls -ld`** or **`stat`** command[42] it is indicated by `d rwx rw`**`s`** `rwx` where s replaces the second x.
sticky	oval-sc:EntityItemBoolType	0..1	false	Indicates that the users can delete each other's files in this directory, when said directory is writable by those users. For the output of the **`ls -ld`** or **`stat`** command[43] it is indicated by `d rwx rwx rw`**`t`** where t replaces the final x for a directory.
uread	oval-sc:EntityItemBoolType	0..1	false	Indicates the owner (user owner) of the file can read this file, or if a directory, read the directory contents. For the output of the **`ls -l`** or `stat` command[44] it is indicated by `d `**`r`**`wx rwx rwx`.
uwrite	oval-sc:EntityItemBoolType	0..1	false	Indicates the owner (user owner) of the file can write to this file, or if a directory, write to the directory. For the output of the **`ls -l`** or **`stat`** command[45] it is indicated by `d r`**`w`**`x rwx rwx`.
uexec	oval-sc:EntityItemBoolType	0..1	false	Indicates the owner

[42] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[43] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[44] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[45] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html

				(user owner) of the file can execute it or, if a directory, change into the directory. For the output of the **`ls -l`** command[46] it is indicated by `d rw`**`x`** `rwx rwx`.
gread	oval-sc:EntityItemBoolType	0..1	false	Indicates the group owner of the file can read this file, or if a directory, read the directory contents. For the output of the **`ls -l`** command[47] it is indicated by `d rwx` **`r`**`wx rwx`.
gwrite	oval-sc:EntityItemBoolType	0..1	false	Indicates the group owner of the file can write to this file, or if a directory, write to the directory. For the output of the **`ls -l`** command[48] it is indicated by `d rwx r`**`w`**`x rwx`.
gexec	oval-sc:EntityItemBoolType	0..1	false	Indicates the group owner of the file can execute it or, if a directory, change into the directory. For the output of the **`ls -l`** command[49] it is indicated by `d rwx rw`**`x`** `rwx`.
oread	oval-sc:EntityItemBoolType	0..1	false	Indicates that all other users can read this file, or if a directory,

[46] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[47] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[48] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[49] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html

				read the directory contents. For the output of the **`ls -l`** command[50] it is indicated by `d rwx rwx` **`r`**`wx`.
owrite	oval-sc:EntityItemBoolType	0..1	false	Indicates that all other users can write to this file, or if a directory, write to the directory. For the output of the **`ls -l`** command[51] it is indicated by `d rwx rwx r`**`w`**`x`.
oexec	oval-sc:EntityItemBoolType	0..1	false	Indicates that all other users can execute the file or, if a directory, change into the directory. For the output of the **`ls -l`** command[52] it is indicated by `d rwx rwx rw`**`x`**.
has_extended_acl	oval-sc:EntityItemBoolType	0..1	false	Indicates the file or directory has ACL permissions[53] applied to it. For the output of the **`ls -l`** or `stat` commands is it indicated by a plus sign (+) appended to the end of the `d rwx rwx rwx` string[54] as in `d rwx rwx rwx` **+**. If a system supports

[50] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[51] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[52] For more information about the different types in the `ls -l` command see http://www.hackinglinuxexposed.com/articles/20030417.html
[53] For more information see http://www.vanemery.com/Linux/ACL/linux-acl.html or http://www.softpanorama.info/Commercial_linuxes/linux_acl.shtml
[54] For more information see http://www.vanemery.com/Linux/ACL/linux-acl.html

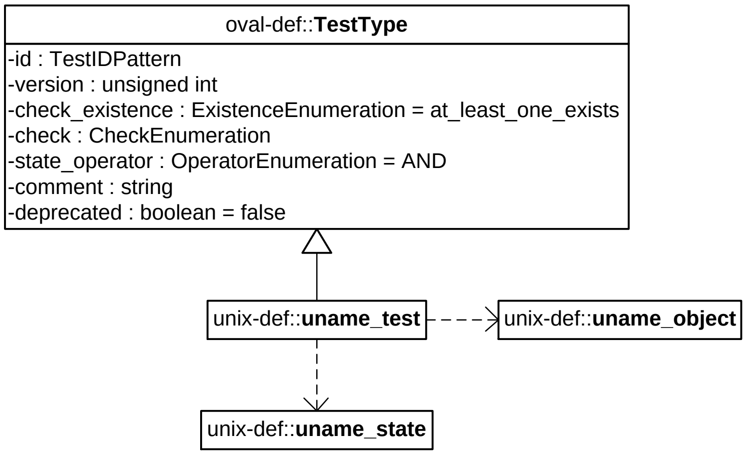
				ACLs and the file or directory doesn't have an ACL, or it matches the standard UNIX permissions, the entity will have a status of *'exists'* and a value of *'false'*. If the system supports ACLs and the file or directory has an ACL, the entity will have a status of *'exists'* and a value of *'true'*. Lastly, if a system doesn't support ACLs, the entity will have a status of *'does not exist'*. If the file or directory doesn't have an ACL, or it matches the standard UNIX permissions, the value with be *false*. Otherwise if a file or directory has an ACL, the value will be *true*.

2.12. unix-def:uname_test

The `uname_test` is used to make assertions about information associated with the hardware the UNIX-based machine is running on[55]. The `uname_test` MUST reference one `uname_object` and

[55] For more information see http://ss64.com/bash/uname.html

zero or more `uname_states`.

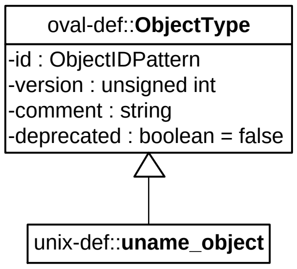

oval-def::**TestType**
-id : TestIDPattern
-version : unsigned int
-check_existence : ExistenceEnumeration = at_least_one_exists
-check : CheckEnumeration
-state_operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::**uname_test**
unix-def::**uname_object**
unix-def::**uname_state**

2.12.1. Known Supported Platforms

- Red Hat Enterprise Linux 5
- Mac OSX 10.6
- Solaris 10

2.13. unix-def:uname_object

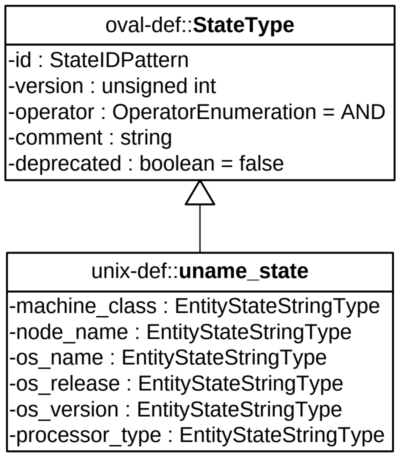
The `uname_object` construct defines the system information[56] that should be collected and represented as `uname_items`. Since there is only one object relating to system information (the system as a whole), there are no child entities defined for this object, so it is considered empty.



2.14. unix-def:uname_state

The `uname_state` construct is used by a `uname_test` to specify system information[57] on UNIX platforms. In getting information about a specific field, a system administrator can use the **`uname`** command or system call[58].

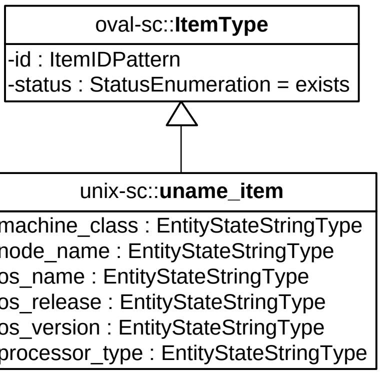
[56] For more information see http://ss64.com/bash/uname.html
[57] For more information about the command line options of the **`uname`** command see http://ss64.com/bash/uname.html
[58] For more information about the **`uname`** system call see http://linux.die.net/man/2/uname


oval-def::**StateType**
-id : StateIDPattern
-version : unsigned int
-operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::**uname_state**
-machine_class : EntityStateStringType
-node_name : EntityStateStringType
-os_name : EntityStateStringType
-os_release : EntityStateStringType
-os_version : EntityStateStringType
-processor_type : EntityStateStringType

Property	Type	Multiplicity	Nillable	Description
machine_class	oval-def: EntityStateStringType	0..1	false	This property specifies a machine hardware name. This corresponds to the command **`uname -m`**.
node_name	oval-def: EntityStateStringType	0..1	false	This property specifies a host name. This corresponds to the command **`uname -n`**.
os_name	oval-def: EntityStateStringType	0..1	false	This property specifies an operating system name. This corresponds to the command **`uname -s`**.
os_release	oval-def: EntityStateStringType	0..1	false	This property specifies a build version. This corresponds to the command **`uname -r`**.
os_version	oval-def: EntityStateStringType	0..1	false	This property specifies an operating system version. This corresponds to the command **`uname -v`**.
processor_type	oval-def: EntityStateStringType	0..*	false	This property specifies a processor type. This corresponds to the command **`uname -p`**.

2.15. unix-sc:uname_item

The `uname_item` construct specifies system information about UNIX platforms[59]. In getting information about a specific field, a system administrator can use the **`uname`** command or system call[60].


oval-sc::ItemType
-id : ItemIDPattern
-status : StatusEnumeration = exists
unix-sc::uname_item
-machine_class : EntityStateStringType
-node_name : EntityStateStringType
-os_name : EntityStateStringType
-os_release : EntityStateStringType
-os_version : EntityStateStringType
-processor_type : EntityStateStringType

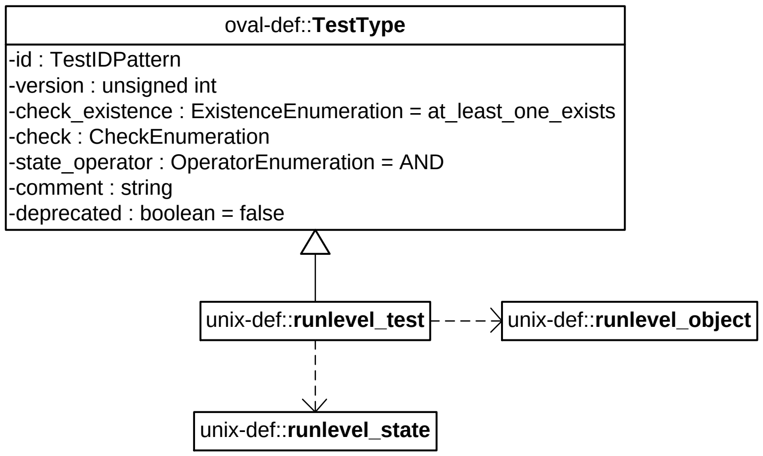
Property	Type	Multiplicity	Nillable	Description
machine_class	oval-sc: EntityItemStringType	0..1	false	This property specifies a machine hardware name. This corresponds to the command **`uname -m`**.
node_name	oval-sc: EntityItemStringType	0..1	false	This property specifies a host name. This corresponds to the command **`uname -n`**.
os_name	oval-sc: EntityItemStringType	0..1	false	This property specifies an operating system name. This corresponds to the command **`uname -s`**.
os_release	oval-sc: EntityItemStringType	0..1	false	This property specifies a build version. This corresponds to the command **`uname -r`**.
os_version	oval-sc: EntityItemStringType	0..1	false	This property specifies an operating system version. This corresponds to the command **`uname -v`**.
processor_type	oval-sc: EntityItemStringType	0..*	false	This property specifies a processor type. This corresponds to the

[59] For more information about the command line options of the `uname` command see http://ss64.com/bash/uname.html
[60] For more information about the **`uname`** system call see http://linux.die.net/man/2/uname

				command **`uname -p`**.

2.7 unix-def:runlevel_test

The `runlevel_test` is used to make assertions about the information of which runlevel specified services are scheduled to exist at. A runlevel is defined as a software configuration of the system that allows only a selected group of processes to exist[61]. To get the runlevel, run the **`init`** command, or use the **`chkconfig --list`** command, which lists the services and runlevels that they can run at[62]. A system administrator must be logged on as root and have root in its own shell (via the commands **`su root`** followed by **`su -`**) or he will get the "command not found" message. The `runlevel_test` MUST reference one `runlevel_object` and zero or more `runlevel_states`.


oval-def::TestType
-id : TestIDPattern
-version : unsigned int
-check_existence : ExistenceEnumeration = at_least_one_exists
-check : CheckEnumeration
-state_operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::runlevel_test
unix-def::runlevel_object
unix-def::runlevel_state

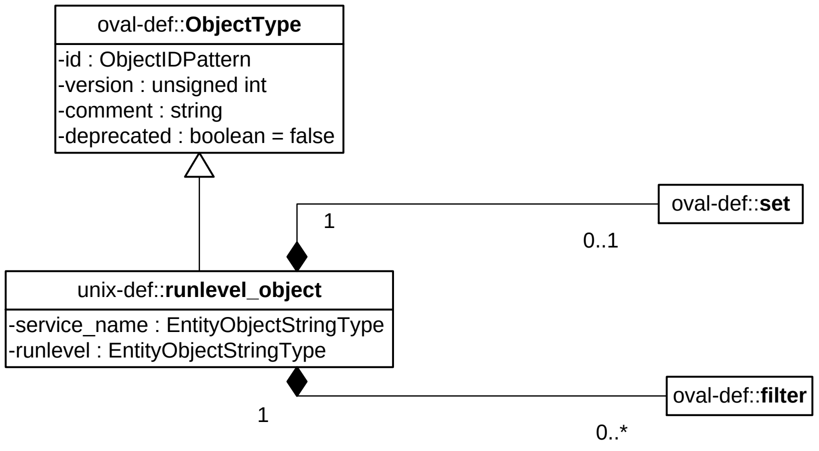
2.7.1 Known Supported Platforms

- Red Hat Enterprise Linux 5
- Mac OSX 10.6
- Solaris 10

2.8 unix-def:runlevel _object

The `runlevel_object` construct defines the set of services/runlevel combinations whose associated system state information should be collected and represented as `runlevel_items`. One can use the **`chkconfig –list`** command to obtain the list of services and the runlevels they can run on[63].

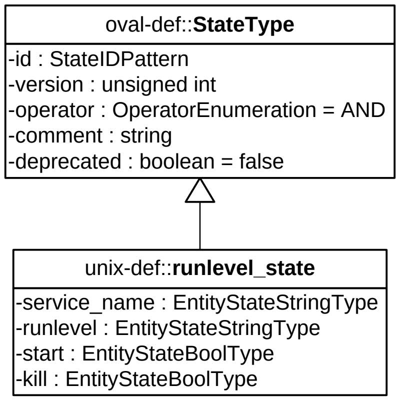
[61] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?init+8
[62] For more information see http://linux.die.net/man/8/chkconfig
[63] For more information see http://linux.die.net/man/8/chkconfig. You must be logged in as root AND have root in its own shell to use the command (via **`su root`** followed by **`su –`**) or it will return "command not found."


oval-def::ObjectType
-id : ObjectIDPattern
-version : unsigned int
-comment : string
-deprecated : boolean = false
oval-def::set
1
0..1
unix-def::runlevel_object
-service_name : EntityObjectStringType
-runlevel : EntityObjectStringType
oval-def::filter
1
0..*

Property	Type	Multiplicity	Nillable	Description
set	oval-def:set	0..1	false	Enables the expression of complex `runlevel_objects` that are the result of logically combining and filtering the `runlevel_items` that are identified by one or more `runlevel_objects`. Please see the OVAL Language Specification for additional information.
service_name	oval-def: EntityObjectStringType	0..1	false	The name associated with a service. This name is usually the filename of the script file located in the /etc/init.d directory.
runlevel	oval-def: EntityObjectStringType	0..1	false	The system runlevel to evaluate. A runlevel is defined as a software configuration of the system that allows only a selected group of processes to exist.
filter	oval-def:filter	0..*	false	Allows for the explicit inclusion or exclusion of `file_items` from the set of `file_items` collected by a `file_object`. Please see the OVAL Language Specification [2] for additional information.

2.9 unix-def: runlevel_state

The `runlevel_state` construct is used by a `runlevel_test` to specify the runlevel information associated with services that should be checked on file systems that are supported by UNIX platforms. One can use the **`chkconfig -list`** command to obtain the list of services and the runlevels they can run on[64].

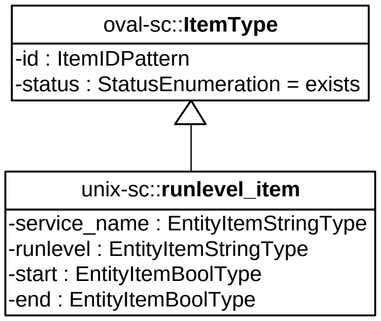

oval-def::StateType
-id : StateIDPattern
-version : unsigned int
-operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::runlevel_state
-service_name : EntityStateStringType
-runlevel : EntityStateStringType
-start : EntityStateBoolType
-kill : EntityStateBoolType

Property	Type	Multiplicity	Nillable	Description
service_name	oval-def:EntityStateStringType	0..1	false	The name associated with a service. This name is usually the filename of the script file located in the /etc/init.d directory.
runlevel	oval-def:EntityStateStringType	0..1	false	The system runlevel to evaluate. A runlevel is defined as a software configuration of the system that allows only a selected group of processes to exist.
start	oval-def:EntityStateBoolType	0..1	false	A process is scheduled to be spawned at the specified runlevel.
kill	oval-def:EntityStateBoolType	0..1	false	A process is scheduled to be killed at the specified runlevel.

[64] For more information see http://linux.die.net/man/8/chkconfig. You must be logged in as root AND have root in its own shell to use the command (via **`su root`** followed by **`su -`**) or it will return "command not found."

2.10 unix-sc:runlevel_item

The `runlevel_item` construct defines the system state information associated with files and directories on file systems supported by the UNIX platform. One can use the **`chkconfig -list`** command to obtain the list of services and the runlevels they can run on[65].


oval-sc::ItemType
-id : ItemIDPattern
-status : StatusEnumeration = exists
unix-sc::runlevel_item
-service_name : EntityItemStringType
-runlevel : EntityItemStringType
-start : EntityItemBoolType
-end : EntityItemBoolType

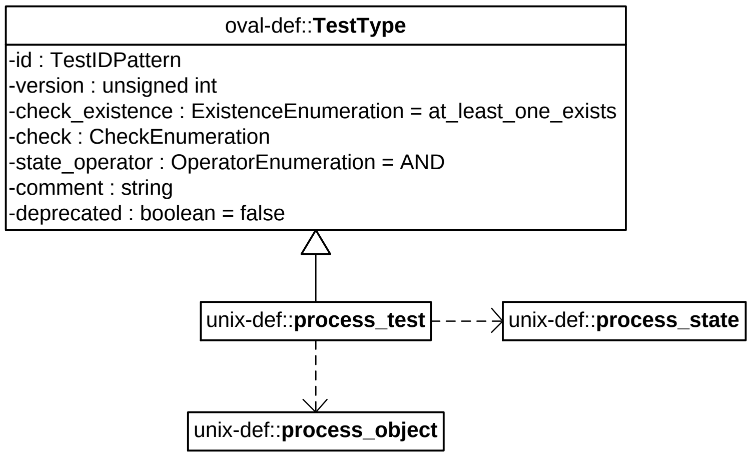
Property	Type	Multiplicity	Nillable	Description
service_name	oval-sc:EntityItemStringType	0..1	false	The name associated with a service. This name is usually the filename of the script file located in the /etc/init.d directory.
runlevel	oval-sc:EntityItemStringType	0..1	false	The system runlevel to evaluate. A runlevel is defined as a software configuration of the system that allows only a selected group of processes to exist.
start	oval-sc:EntityItemBoolType	0..1	false	A process is scheduled to be spawned at the specified runlevel.
kill	oval-sc:EntityItemBoolType	0..1	false	A process is scheduled to be killed at the specified runlevel.

2.11 unix-def:process_test

The `process_test` is used to make assertions about processes on a UNIX system, especially information given as output via the **`ps`** command[66]. Notice that the ps command may have different

[65] For more information see http://linux.die.net/man/8/chkconfig. You must be logged in as root AND have root in its own shell to use the command (via **`su root`** followed by **`su -`**) or it will return "command not found."
[66] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?ps

implementations across platforms depending on the flags and outputs set by the vendor[67]. The `process_test` MUST reference one `process_object` and zero or more `process_states`.

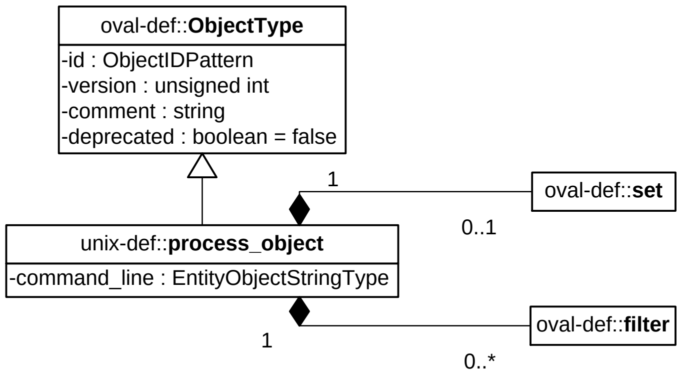

oval-def::TestType
-id : TestIDPattern
-version : unsigned int
-check_existence : ExistenceEnumeration = at_least_one_exists
-check : CheckEnumeration
-state_operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::process_test
unix-def::process_state
unix-def::process_object

2.11.1 Known Supported Platforms

- Red Hat Enterprise Linux 5
- Mac OSX 10.6
- Solaris 10

2.12 unix-def:process_object

The `process_object` construct defines the set of processes whose associated information should be collected and represented as `process_items`[68].



Property	Type	Multiplicity	Nillable	Description

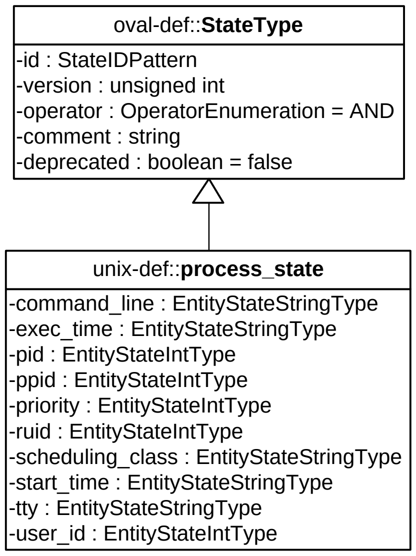
[67] For more information see http://kb.iu.edu/data/afnv.html
[68] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?ps

set	oval-def:set	0..1	false	Enables the expression of complex `process_objects` that are the result of logically combining and filtering the `process_items` that are identified by one or more `process_objects`. Please see the OVAL Language Specification for additional information.
command	oval-def: EntityObjectStringType	0..1	false	Specifies which command/program name to check.
filter	oval-def:filter	0..*	false	Allows for the explicit inclusion or exclusion of `process_items` from the set of `process_items` collected by a `process_object`. Please see the OVAL Language Specification [2] for additional information.

2.13 unix-def:process_state

The `process_state` construct is used by a `process_test` to specify information about processes on UNIX platforms. To get this information an administrator can use the **`ps`** command[69] or obtain information from /proc/<pid>/psinfo, where <pid> is the process identifier of an individual process[70]. An alternate name and command to access (with minimum effort) is provided for convenience as it relates to **`ps`**'s output.

[69] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?ps
[70] For more information about obtaining the ps output from system calls see http://www.mitchr.me/SS/exampleCode/AUPG/solaris_ps.c.html for the source code. The line sprintf(fileToOpen, "/proc/%s/psinfo", dep->d_name) is of particular interest. Please note that the psinfo part of the process information path may vary for different UNIX systems. For example, in CentOS, status is used instead of psinfo.


oval-def::StateType
-id : StateIDPattern
-version : unsigned int
-operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::process_state
-command_line : EntityStateStringType
-exec_time : EntityStateStringType
-pid : EntityStateIntType
-ppid : EntityStateIntType
-priority : EntityStateIntType
-ruid : EntityStateIntType
-scheduling_class : EntityStateStringType
-start_time : EntityStateStringType
-tty : EntityStateStringType
-user_id : EntityStateIntType

Property	Type	Multiplicity	Nillable	Description
command	oval-def:EntityStateStringType	0..1	false	Alternate name: COMMAND. The command property specifies the command/program name to check. Accessible via **`ps`**.
exec_time	oval-def:EntityStateStringType	0..1	false	Alternate name: TIME. This is the cumulative CPU time, formatted in [DD-]HH:MM:SS where DD is the number of days when execution time is 24 hours or more. This can be adjusted implicitly via the **`nice`** command or nice() system call. Accessible via **`ps`**.
pid	oval-def:EntityStateIntType	0..1	false	Alternate name: PID. This is the process ID of the process. Accessible via **`ps`**.
ppid	oval-def:EntityStateIntType	0..1	false	Alternate name: PPID. This is the process ID of the process's parent process.

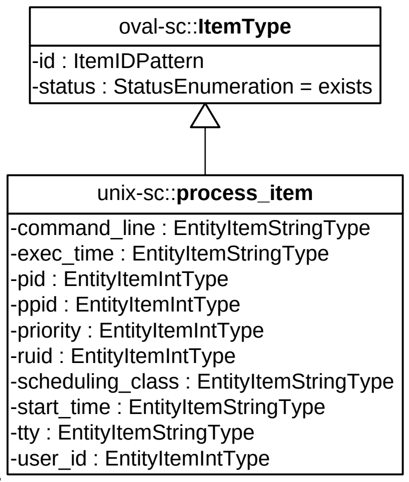
				Accessible via **`ps -f`**.
priority	oval-def:EntityStateIntType	0..1	false	Alternate name: RTPRIO. This is the scheduling priority with which the process runs. This can be adjusted with the **`nice`** command or nice() system call. Accessed via **`ps -o rtprio,*`** where **`*`** is any combination of pids, commands, or fields that could be specified for clarification.
ruid	oval-def:EntityStateIntType	0..1	false	Alternate name: RUID. This is the real user id which represents the user who has created the process. Accessed via **`ps -o ruid,*`** where `*` is any combination of pids, commands, or fields that could be specified for clarification.
scheduling_class	oval-def:EntityStateStringType	0..1	false	Alternate name: CLS. A platform specific characteristic maintained by the scheduler: RT (real-time), TS (timeshare), FF (fifo), SYS (system), etc. Accessed via **`ps -o cls,*`** where `*` is any combination of pids, commands, or fields that could be specified for clarification.
start_time	oval-def:EntityStateStringType	0..1	false	Alternate name: STARTED or START (abbreviated). This is the time of day the process started, formatted in

				HH:MM:SS (or HH:MM) if the same day the process started or formatted as MMM_DD (Ex.: Feb_5) if process started the previous day or further in the past. The best way to get this information is to use **`ps -o start,*`** for the HH:MM:SS format.
tty	oval-def:EntityStateStringType	0..1	false	Alternate name: TTY. This is the TTY on which the process was started, if applicable. Accessible via **`ps`**.
user_id	oval-def:EntityStateIntType	0..1	false	Alternate names: UID (sometimes—works under **`ps -l`** but NOT **`ps -f`**). This is the effective user id (a number, not a string) which represents the actual privileges of the process. Best accestable via **`ps -l.`**

2.14 unix-sc:process_item

The `process_item` construct defines the information associated with processes on file systems supported by the UNIX platform. To get this information an administrator can use the **`ps`** command[71] or obtain information from /proc/<pid>/psinfo, where <pid> is the process identifier of an individual process[72]. An alternate name and command to access (with minimum effort) is provided for convenience as it relates to **`ps`**'s output.

[71] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?ps
[72] For more information about obtaining the ps output from system calls see http://www.mitchr.me/SS/exampleCode/AUPG/solaris_ps.c.html for the source code. The line sprintf(fileToOpen, "/proc/%s/psinfo", dep->d_name) is of particular interest. Please note that the psinfo part of the process information path may vary for different UNIX systems. For example, in CentOS, status is used instead of psinfo.


oval-sc::ItemType
-id : ItemIDPattern
-status : StatusEnumeration = exists
unix-sc::process_item
-command_line : EntityItemStringType
-exec_time : EntityItemStringType
-pid : EntityItemIntType
-ppid : EntityItemIntType
-priority : EntityItemIntType
-ruid : EntityItemIntType
-scheduling_class : EntityItemStringType
-start_time : EntityItemStringType
-tty : EntityItemStringType
-user_id : EntityItemIntType

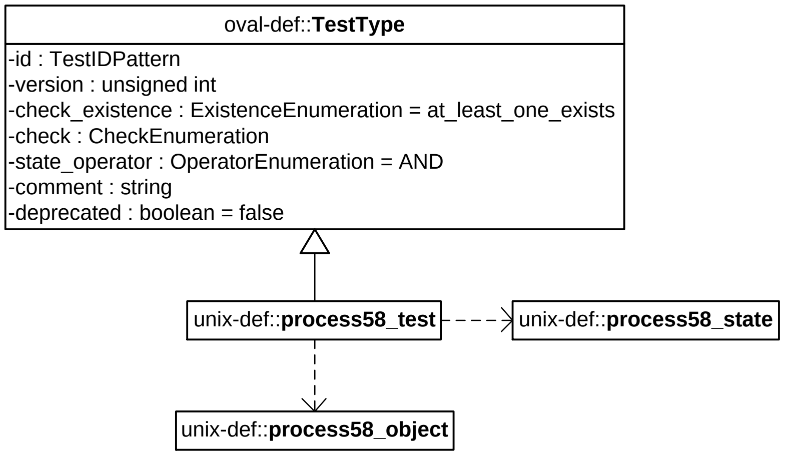
Property	Type	Multiplicity	Nillable	Description
command	oval-sc:EntityItemStringType	0..1	false	Alternate name: COMMAND. The command element specifies the command/program name to check. Accessible via **`ps`**.
exec_time	oval-sc:EntityItemStringType	0..1	false	Alternate name: TIME. This is the cumulative CPU time, formatted in [DD-]HH:MM:SS where DD is the number of days when execution time is 24 hours or more. This can be adjusted implicitly via the `nice` command. Accessible via **`ps`**.
pid	oval-sc:EntityItemIntType	0..1	false	Alternate name: PID. This is the process ID of the process. Accessible via **`ps`**.
ppid	oval-sc:EntityItemIntType	0..1	false	Alternate name: PPID. This is the process ID of the process's parent process. Accessible via **`ps -f`**.
priority	oval-sc: EntityItemIntType	0..1	false	Alternate name: RTPRIO. This is the

				scheduling priority with which the process runs. This can be adjusted with the nice command or nice() system call. Accessed via **`ps -o rtprio,*`** where **`*`** is any combination of pids, commands, or fields that could be specified for clarification.
ruid	oval-sc: EntityItemIntType	0..1	false	Alternate name: RUID. This is the real user id which represents the user who has created the process. Accessed via **`ps -o ruid,*`** where `*` is any combination of pids, commands, or fields that could be specified for clarification.
scheduling_class	oval-sc: EntityItemStringType	0..1	false	Alternate name: CLS. A platform specific characteristic maintained by the scheduler: RT (real-time), TS (timeshare), FF (fifo), SYS (system), etc. Accessed via **`ps -o cls,*`** where **`*`** is any combination of pids, commands, or fields that could be specified for clarification.
start_time	oval-sc: EntityItemStringType	0..1	false	Alternate name: STARTED or START (abbreviated). This is the time of day the process started, formatted in HH:MM:SS (or HH:MM) if the same day the process

				started or formatted as MMM_DD (Ex.: Feb_5) if process started the previous day or further in the past. The best way to get this information is to use **`ps –o start,*`** for the HH:MM:SS format.
tty	oval-sc: EntityItemStringType	0..1	false	Alternate name: TTY. This is the TTY on which the process was started, if applicable. Accessible via `ps`.
user_id	oval-sc: EntityItemIntType	0..1	false	Alternate names: UID (sometimes—works under **`ps –l`** but NOT **`ps -f`**). This is the effective user id (a number, not a string) which represents the actual privileges of the process. Best accestable via **`ps –l.`**

2.15 unix-def:process58_test

The `process58_test` is used to make assertions about processes on a UNIX system, especially information given as output via the `ps` command[73]. Notice that the **`ps`** command may have different UNIX implementations depending on the flags and outputs set by the vendor[74]. The `process58_test` MUST reference one `process58_object` and zero or more `process58_states`.


oval-def::**TestType**
-id : TestIDPattern
-version : unsigned int
-check_existence : ExistenceEnumeration = at_least_one_exists
-check : CheckEnumeration
-state_operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::**process58_test**
unix-def::**process58_state**
unix-def::**process58_object**

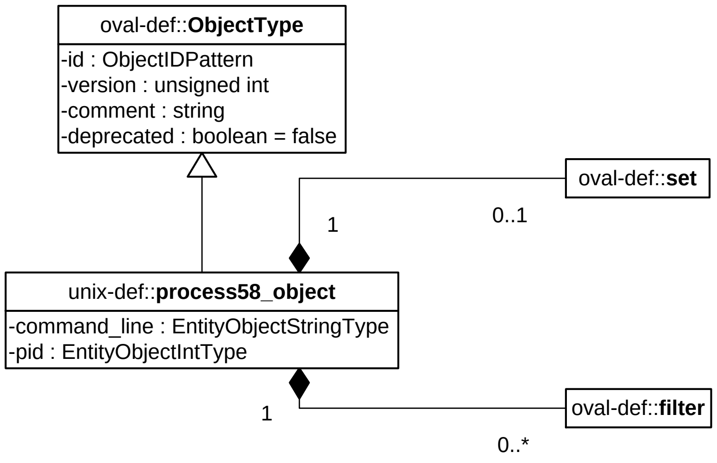
2.15.1 Known Supported Platforms

- Red Hat Enterprise Linux 5
- Mac OSX 10.6
- Solaris 10

2.16 unix-def:process58_object

The `process58_object` construct defines the set of processes, via BOTH the command_line and pid properties, whose associated information should be collected and represented as `process58_items`[75].

[73] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?ps
[74] For more information see http://kb.iu.edu/data/afnv.html
[75] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?ps



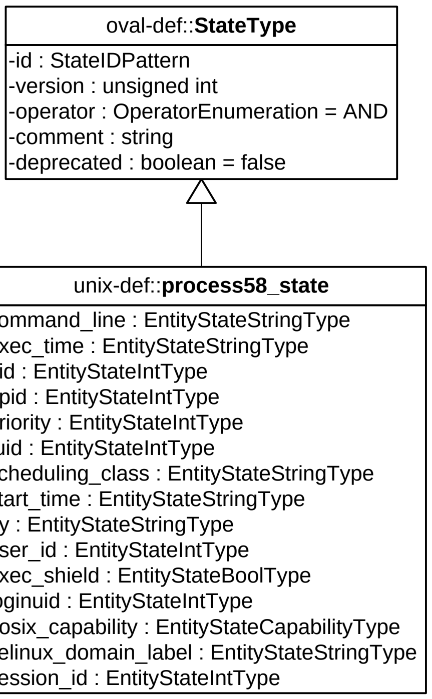
Property	Type	Multiplicity	Nillable	Description
set	oval-def:set	0..1	false	Enables the expression of complex `process58_objects` that are the result of logically combining and filtering the `process58_items` that are identified by one or more `process58_objects`. Please see the OVAL Language Specification for additional information.
command_line	oval-def: EntityObjectStringType	0..1	false	Specifies which command/program name to check.
pid	oval-def: EntityObjectIntType	0..1	false	Alternate name: PID. This is the process ID of the process. Accessible via `ps`.
filter	oval-def:filter	0..*	false	Allows for the explicit inclusion or exclusion of `process58_items` from the set of `process58_items` collected by a `process58_object`. Please see the OVAL Language Specification [2] for additional information.

2.17 unix-def: process58_state

The `process58_state` construct is used by a `process58_test` to specify information about processes on UNIX platforms. To get this information an administrator can use the **`ps`** command[76] or

[76] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?ps

obtain information from /proc/<pid>/psinfo, where <pid> is the process identifier of an individual process[77]. An alternate name and command to access (with minimum effort) is provided for convenience as it relates to **ps**'s output.


oval-def::StateType
-id : StateIDPattern
-version : unsigned int
-operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::process58_state
-command_line : EntityStateStringType
-exec_time : EntityStateStringType
-pid : EntityStateIntType
-ppid : EntityStateIntType
-priority : EntityStateIntType
-ruid : EntityStateIntType
-scheduling_class : EntityStateStringType
-start_time : EntityStateStringType
-tty : EntityStateStringType
-user_id : EntityStateIntType
-exec_shield : EntityStateBoolType
-loginuid : EntityStateIntType
-posix_capability : EntityStateCapabilityType
-selinux_domain_label : EntityStateStringType
-session_id : EntityStateIntType

Property	Type	Multiplicity	Nillable	Description
command	oval-def:EntityStateStringType	0..1	false	Alternate name: COMMAND. The command element specifies the command/program name to check. Accessible via **ps**.
exec_time	oval-def:EntityStateStringType	0..1	false	Alternate name: TIME. This is the cumulative CPU time, formatted in [DD-]HH:MM:SS where DD is the number of days when execution time is 24 hours or more. This

[77] For more information about obtaining the ps output from system calls see http://www.mitchr.me/SS/exampleCode/AUPG/solaris_ps.c.html for the source code. The line sprintf(fileToOpen, "/proc/%s/psinfo", dep->d_name) is of particular interest. Please note that the psinfo part of the process information path may vary for different UNIX systems. For example, in CentOS, status is used instead of psinfo.

				can be adjusted implicitly via the `nice` command. Accessible via **`ps`**.
pid	oval-def:EntityStateIntType	0..1	false	Alternate name: PID. This is the process ID of the process. Accessible via `ps`.
ppid	oval-def:EntityStateIntType	0..1	false	Alternate name: PPID. This is the process ID of the process's parent process. Accessible via **`ps -f`**.
priority	oval-def:EntityStateIntType	0..1	false	Alternate name: RTPRIO? This is the scheduling priority with which the process runs. This can be adjusted with the **`nice`** command or nice() system call. Accessed via **`ps -o rtprio,*`** where `*` is any combination of pids, commands, or fields that could be specified for clarification.
ruid	oval-def:EntityStateIntType	0..1	false	Alternate name: RUID. This is the real user id which represents the user who has created the process. Accessed via **`ps -o ruid,*`** where **`*`** is any combination of pids, commands, or fields that could be specified for clarification.
scheduling_class	oval-def:EntityStateStringType	0..1	false	Alternate name: CLS. A platform specific characteristic maintained by the scheduler: RT (real-time), TS (timeshare), FF (fifo), SYS (system), etc. Accessed via **`ps`**

				-o cls,* where * is any combination of pids, commands, or fields that could be specified for clarification.
start_time	oval-def:EntityStateStringType	0..1	false	Alternate name: STARTED or START (abbreviated). This is the time of day the process started, formatted as HH:MM:SS (or HH:MM) if the same day the process started or formatted as MMM_DD (Ex.: Feb_5) if process started the previous day or further in the past. The best way to get this information is to use **ps -o start,*** for the HH:MM:SS format.
tty	oval-def:EntityStateStringType	0..1	false	Alternate name: TTY. This is the TTY on which the process was started, if applicable. Accessible via ps.
user_id	oval-def:EntityStateIntType	0..1	false	Alternate names: UID (sometimes—works under **ps -l** but NOT **ps -f**). This is the effective user id (a number, not a string) which represents the actual privileges of the process. Best accestable via **ps -l.**
exec_shield	oval-def:EntityStateBoolType	0..1	false	A boolean that when true would indicate that ExecShield is enabled for the

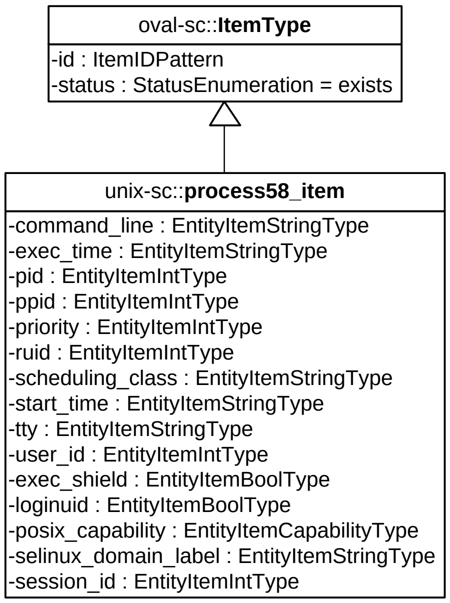
				process.
loginuid	oval-def:EntityStateIntType	0..1	false	The loginuid shows which account a user gained access to the system with. The /proc/XXXX/loginuid shows this value. If the value is -1, cast as an unsigned int, the loginuid was unset[78].
posix_capability	unix-def: EntityStateCapabilityType	0..1	false	An effective capability associated with the process. This can be accessed via proc/<pid>/status under the value, capeff.
selinux_domain_label	oval-def:EntityStateStringType	0..1	false	An selinux domain (or type) label associated with the process. This domain label corresponds to the type specified via the **`secon`** command or the getpidcon() system call[79].
session_id	oval-def:EntityStateIntType	0..1	false	Alternate name: SID The session ID of the process. If the values of session_id and pid match, then this process is also a session leader[80]. Accessed via **`ps -o sid,*`** where **`*`** is any combination of pids, commands, or fields that could be

[78] For more information see http://linux.die.net/man/3/audit_getloginuid
[79] For more information see http://linux.die.net/man/3/getpidcon for the system call, http://linux.die.net/man/1/secon for the command, and http://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-selinux.html for more information. Note that there is NO DIFFERENCE between a domain and a type—see http://docs.fedoraproject.org/en-US/Fedora/13/html/SELinux_FAQ/
[80] For more information see http://www.informit.com/articles/article.aspx?p=397655&seqNum=6 or http://unix.stackexchange.com/questions/18166/what-are-session-leaders-in-ps

				specified for clarification.

2.18 unix-sc:process58_item

The `process58_item` construct defines the information associated with processes on file systems supported by the UNIX platform. To get this information an administrator can use the **`ps`** command[81] or obtain information from /proc/<pid>/psinfo, where <pid> is process identifier of an individual process[82]. An alternate name and command to access (with minimum effort) is provided for convenience as it relates to **`ps`**'s output.


oval-sc::ItemType
-id : ItemIDPattern
-status : StatusEnumeration = exists
unix-sc::process58_item
-command_line : EntityItemStringType
-exec_time : EntityItemStringType
-pid : EntityItemIntType
-ppid : EntityItemIntType
-priority : EntityItemIntType
-ruid : EntityItemIntType
-scheduling_class : EntityItemStringType
-start_time : EntityItemStringType
-tty : EntityItemStringType
-user_id : EntityItemIntType
-exec_shield : EntityItemBoolType
-loginuid : EntityItemBoolType
-posix_capability : EntityItemCapabilityType
-selinux_domain_label : EntityItemStringType
-session_id : EntityItemIntType

Property	Type	Multiplicity	Nillable	Description
command	oval-sc:EntityItemStringType	0..1	false	Alternate name: COMMAND. The command element specifies the command/program name to check. Accessible via **`ps`**.
exec_time	oval-sc:EntityItemStringType	0..1	false	Alternate name: TIME. This is the cumulative

[81] For more information see http://unixhelp.ed.ac.uk/CGI/man-cgi?ps
[82] For more information about obtaining the ps output from system calls see http://www.mitchr.me/SS/exampleCode/AUPG/solaris_ps.c.html for the source code. The line sprintf(fileToOpen, "/proc/%s/psinfo", dep->d_name) is of particular interest. Please note that the psinfo part of the process information path may vary for different UNIX systems. For example, in CentOS, status is used instead of psinfo.

				CPU time, formatted in [DD-]HH:MM:SS where DD is the number of days when execution time is 24 hours or more. This can be adjusted implicitly via the `nice` command. Accessible via **`ps`**.
pid	oval-sc:EntityItemIntType	0..1	false	Alternate name: PID. This is the process ID of the process. Accessible via `ps`.
ppid	oval-sc:EntityItemIntType	0..1	false	Alternate name: PPID. This is the process ID of the process's parent process. Accessible via **`ps -f`**.
priority	oval-sc: EntityItemIntType	0..1	false	Alternate name: RTPRIO? This is the scheduling priority with which the process runs. This can be adjusted with the **`nice`** command or nice() system call. Accessed via **`ps -o rtprio,*`** where `*` is any combination of pids, commands, or fields that could be specified for clarification.
ruid	oval-sc: EntityItemIntType	0..1	false	Alternate name: RUID. This is the real user id which represents the user who has created the process. Accessed via **`ps -o ruid,*`** where **`*`** is any combination of pids, commands, or fields that could be specified for clarification.
scheduling_class	oval-sc: EntityItemStringType	0..1	false	Alternate name: CLS. A platform specific

				characteristic maintained by the scheduler: RT (real-time), TS (timeshare), FF (fifo), SYS (system), etc. Accessed via **`ps -o cls,*`** where `*` is any combination of pids, commands, or fields that could be specified for clarification.
start_time	oval-sc: EntityItemStringType	0..1	false	Alternate name: STARTED or START (abbreviated). This is the time of day the process started, formatted as HH:MM:SS (or HH:MM) if the same day the process started or formatted as MMM_DD (Ex.: Feb_5) if process started the previous day or further in the past. The best way to get this information is to use **`ps -o start,*`** for the HH:MM:SS format.
tty	oval-sc: EntityItemStringType	0..1	false	Alternate name: TTY. This is the TTY on which the process was started, if applicable. Accessible via `ps`.
user_id	oval-sc: EntityItemIntType	0..1	false	Alternate names: UID (sometimes—works under `ps -l` but NOT `ps -f`). This is the effective user id (a number, not a string) which represents the actual privileges of the process. Best

				accestable via `ps -l`.
exec_shield	oval-def:EntityStateBoolType	0..1	false	A boolean that when true would indicate that ExecShield is enabled for the process.
loginuid	oval-def:EntityStateIntType	0..1	false	The loginuid shows which account a user gained access to the system with. The /proc/XXXX/loginuid shows this value. If the value is -1, cast as an unsigned int, the loginuid was unset[83].
posix_capability	unix-def: EntityStateCapabilityType	0..1	false	An effective capability associated with the process. This can be accessed via proc/<pid>/status under the value, capeff.
selinux_domain_label	oval-def:EntityStateStringType	0..1	false	An selinux domain (or type) label associated with the process. This domain label corresponds to the type specified via the `secon` command or the getpidcon() system call[84].
session_id	oval-def:EntityStateIntType	0..1	false	Alternate name: SID The session ID of the process. If the values of session_id and pid match, then this process is also a session leader[85].

[83] For more information see http://linux.die.net/man/3/audit_getloginuid
[84] For more information see http://linux.die.net/man/3/getpidcon for the system call, http://linux.die.net/man/1/secon for the command, and http://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-selinux.html for more information. Note that there is NO DIFFERENCE between a domain and a type—see http://docs.fedoraproject.org/en-US/Fedora/13/html/SELinux_FAQ/
[85] For more information see http://www.informit.com/articles/article.aspx?p=397655&seqNum=6 or http://unix.stackexchange.com/questions/18166/what-are-session-leaders-in-ps

				Accessed via **ps -o sid,*** where * is any combination of pids, commands, or fields that could be specified for clarification.

2.19. unix-def:EntityStateCapabilityType

The `EntityStateCapabilityType` defines the values that describe POSIX capability[86] types associated with a process service on UNIX systems. This list is based off the values defined in linux/include/linux/capability.h[87].

Enumeration Value	Description
CAP_CHOWN	Defined as 0 in capability.h. In a system with the [_POSIX_CHOWN_RESTRICTED] option defined, this overrides the restriction of changing file ownership and group ownership.
CAP_DAC_OVERRIDE	Defined as 1 in capability.h. Override all DAC access, including ACL execute access if POSIX_ACL] is defined. Excluding DAC access covered by CAP_LINUXIMMUTABLE.
CAP_DAC_READ_SEARCH	Defined as 2 in capability.h. Overrides all DAC restrictions regarding read and search on files and directories, including ACL restrictions if POSIX_ACL] is defined. Excluding DAC access covered by CAP_LINUXIMMUTABLE.
CAP_FOWNER	Defined as 3 in capability.h. Overrides all restrictions about allowed operations on files, where file owner ID must be equal to the user ID, except where CAP_FSETID is applicable. It doesn't override MAC and DAC restrictions.
CAP_FSETID	Defined as 4 in capability.h. Overrides the following restrictions that the effective user ID shall match the file owner ID when setting the S_ISUID and S_ISGID bits on that file; that the effective group ID (or one of the supplementary group IDs) shall match the file owner ID when setting the S_ISGID bit on that file; that the S_ISUID and S_ISGID bits are cleared on successful return from chown(2) (not implemented).
CAP_KILL	Defined as 5 in capability.h. Overrides the restriction that the real or effective user ID of a process sending a signal must match the real or effective user ID of the process receiving the signal.
CAP_SETGID	Defined as 6 in capability.h. Allows setgid(2) manipulation,

[86] For more information see http://www.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.2/capfaq-0.2.txt
[87] For more information see http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/capability.h

	setgroups(2), and forged gids on socket credentials passing.
CAP_SETUID	Defined as 7 in capability.h. Allows set*uid(2) manipulation (including fsuid) and forged pids on socket credentials passing.
CAP_SETPCAP	Defined as 8 in capability.h. Linux-specific capabilities: Transfer any capability in your permitted set to any pid, remove any capability in your permitted set from any pid.
CAP_LINUX_IMMUTABLE	Defined as 9 in capability.h. Allow modification of S_IMMUTABLE and S_APPEND file attributes.
CAP_NET_BIND_SERVICE	Defined as 10 in capability.h. Allows binding to TCP/UDP sockets below 1024 and binding to ATM VCIs below 32
CAP_NET_BROADCAST	Defined as 11 in capability.h. Allow broadcasting and listening to multicast.
CAP_NET_ADMIN	Defined as 12 in capability.h. Allows certain administrative rights, including interface configuration, administration of IP firewall, masquerading and accouting, and setting dubug option on sockets. The full list can be found in linux/include/linux/capability.h[88].
CAP_NET_RAW	Defined as 13 in capability.h. Allows the use of RAW and PACKET sockets.
CAP_IPC_LOCK	Defined as 14 in capability.h. Allows the locking of shared memory segments and mlock and mlockall (which doesn't really have anything to do with IPC).
CAP_IPC_OWNER	Defined as 15 in capability.h. Overrides IPC ownership checks.
CAP_SYS_MODULE	Defined as 16 in capability.h. Insert and remove kernel modules – modify kernel without limit, and modify cap_bset.
CAP_SYS_RAWIO	Defined as 17 in capability.h. Allow ioperm/iopl access and the sending of USB messages to any device via /proc/bus/usb.
CAP_SYS_CHROOT	Defined as 18 in capability.h. Allows use of chroot().
CAP_SYS_PTRACE	Defined as 19 in capability.h. Allow ptrace() of any process.
CAP_SYS_PACCT	Defined as 20 in capability.h. Allow configuration of process accounting.
CAP_SYS_ADMIN	Defined as 21 in capability.h. Allows for many rights, including configuration of the secure attention key, administration of the random device, examination and configuration of disk quotas, among others. The full list can be found in linux/include/linux/capability.h[89].
CAP_SYS_BOOT	Defined as 22 in capability.h. Allow use of reboot().
CAP_SYS_NICE	Defined as 23 in capability.h. Allows raising priority and setting priority on other (different UID) processes, the use of FIFO and round-robin (realtime) scheduling on own processes and setting the scheduling algorithm used by another process, and setting cpu affinity on other processes.

[88] For more information see http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/capability.h

[89] For more information see http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/capability.h

CAP_SYS_RESOURCE	Defined as 24 in capability.h. Overrides certain limitations, such as resource limits, quota limits, reserved space on ext2 filesystems, among other tasks which are listed in linux/include/linux/capability.h[90].
CAP_SYS_TIME	Defined as 25 in capability.h. Allow manipulation of system clock, irix_stime on mips and setting the real-time clock.
CAP_SYS_TTY_CONFIG	Defined as 26 in capability.h. Allow configuration of tty devices and vhangup() of tty.
CAP_MKNOD	Defined as 27 in capability.h. Allow the privileged aspects of mknod().
CAP_LEASE	Defined as 28 in capability.h. Allow taking of leases on files.
CAP_AUDIT_WRITE	Defined as 29 in capability.h.
CAP_AUDIT_CONTROL	Defined as 30 in capability.h.
CAP_SETFCAP	Defined as 31 in capability.h. NOT supported on all UNIX OSes as many versions of capability.h stop at 30.
CAP_MAC_OVERRIDE	Defined as 32 in capability.h. Override MAC access. The base kernel enforces no MAC policy. An LSM may enforce a MAC policy, and if it does and it chooses to implement capability based overrides of that policy, this is the capability it should use to do so. NOT supported on all UNIX OSes as many versions of capability.h stop at 30.
CAP_MAC_ADMIN	Defined as 33 in capability.h. Allow MAC configuration or state changes. The base kernel requires no MAC configuration. An LSM may enforce a MAC policy, and if it does and it chooses to implement capabilitybased checks on modifications to that policy or the data required to maintain it, this is the capability it should use to do so.
<empty string>	This value indicates that no value has been specified and is permitted here to allow for an empty entity which is associated with error and not collected conditions.

2.20. unix-sc:EntityItemCapabilityType

The `EntityItemCapabilityType` defines the enumeration of values that describe POSIX capability[91] types associated with a process service on UNIX systems. This list is based off the values defined in linux/include/linux/capability.h[92].

Enumeration Value	Description
CAP_CHOWN	Defined as 0 in capability.h. In a system with the [_POSIX_CHOWN_RESTRICTED] option defined, this overrides the restriction of changing file ownership and group ownership.

[90] For more information see http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/capability.h
[91] For more information see http://www.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.2/capfaq-0.2.txt
[92] For more information see http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/capability.h

CAP_DAC_OVERRIDE	Defined as 1 in capability.h. Override all DAC access, including ACL execute access if POSIX_ACL] is defined. Excluding DAC access covered by CAP_LINUXIMMUTABLE.
CAP_DAC_READ_SEARCH	Defined as 2 in capability.h. Overrides all DAC restrictions regarding read and search on files and directories, including ACL restrictions if POSIX_ACL] is defined. Excluding DAC access covered by CAP_LINUXIMMUTABLE.
CAP_FOWNER	Defined as 3 in capability.h. Overrides all restrictions about allowed operations on files, where file owner ID must be equal to the user ID, except where CAP_FSETID is applicable. It doesn't override MAC and DAC restrictions.
CAP_FSETID	Defined as 4 in capability.h. Overrides the following restrictions that the effective user ID shall match the file owner ID when setting the S_ISUID and S_ISGID bits on that file; that the effective group ID (or one of the supplementary group IDs) shall match the file owner ID when setting the S_ISGID bit on that file; that the S_ISUID and S_ISGID bits are cleared on successful return from chown(2) (not implemented).
CAP_KILL	Defined as 5 in capability.h. Overrides the restriction that the real or effective user ID of a process sending a signal must match the real or effective user ID of the process receiving the signal.
CAP_SETGID	Defined as 6 in capability.h. Allows setgid(2) manipulation, setgroups(2), and forged gids on socket credentials passing.
CAP_SETUID	Defined as 7 in capability.h. Allows set*uid(2) manipulation (including fsuid) and forged pids on socket credentials passing.
CAP_SETPCAP	Defined as 8 in capability.h. Linux-specific capabilities: Transfer any capability in your permitted set to any pid, remove any capability in your permitted set from any pid.
CAP_LINUX_IMMUTABLE	Defined as 9 in capability.h. Allow modification of S_IMMUTABLE and S_APPEND file attributes.
CAP_NET_BIND_SERVICE	Defined as 10 in capability.h. Allows binding to TCP/UDP sockets below 1024 and binding to ATM VCIs below 32
CAP_NET_BROADCAST	Defined as 11 in capability.h. Allow broadcasting and listening to multicast.
CAP_NET_ADMIN	Defined as 12 in capability.h. Allows certain administrative rights, including interface configuration, administration of IP firewall, masquerading and accouting, and setting dubug option on sockets. The full list can be found in linux/include/linux/capability.h[93].
CAP_NET_RAW	Defined as 13 in capability.h. Allows the use of RAW and PACKET sockets.
CAP_IPC_LOCK	Defined as 14 in capability.h. Allows the locking of shared memory segments and mlock and mlockall (which doesn't really have anything to do with IPC).

[93] For more information see http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/capability.h

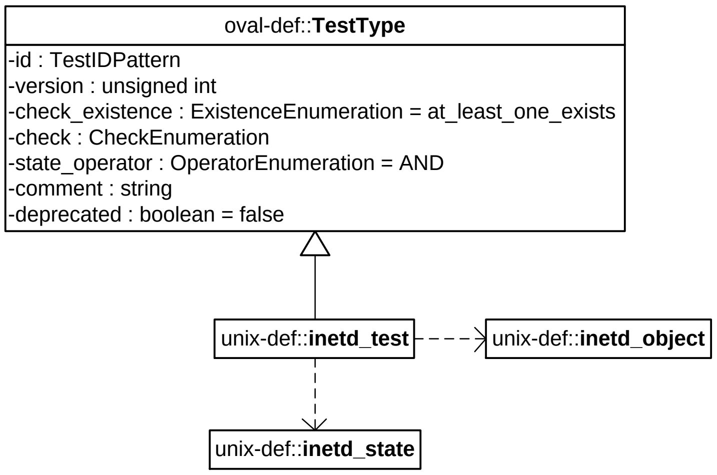
CAP_IPC_OWNER	Defined as 15 in capability.h. Overrides IPC ownership checks.
CAP_SYS_MODULE	Defined as 16 in capability.h. Insert and remove kernel modules – modify kernel without limit, and modify cap_bset.
CAP_SYS_RAWIO	Defined as 17 in capability.h. Allow ioperm/iopl access and the sending of USB messages to any device via /proc/bus/usb.
CAP_SYS_CHROOT	Defined as 18 in capability.h. Allows use of chroot().
CAP_SYS_PTRACE	Defined as 19 in capability.h. Allow ptrace() of any process.
CAP_SYS_PACCT	Defined as 20 in capability.h. Allow configuration of process accounting.
CAP_SYS_ADMIN	Defined as 21 in capability.h. Allows for many rights, including configuration of the secure attention key, administration of the random device, examination and configuration of disk quotas, among others. The full list can be found in linux/include/linux/capability.h[94].
CAP_SYS_BOOT	Defined as 22 in capability.h. Allow use of reboot().
CAP_SYS_NICE	Defined as 23 in capability.h. Allows raising priority and setting priority on other (different UID) processes, the use of FIFO and round-robin (realtime) scheduling on own processes and setting the scheduling algorithm used by another process, and setting cpu affinity on other processes.
CAP_SYS_RESOURCE	Defined as 24 in capability.h. Overrides certain limitations, such as resource limits, quota limits, reserved space on ext2 filesystems, among other tasks which are listed in linux/include/linux/capability.h[95].
CAP_SYS_TIME	Defined as 25 in capability.h. Allow manipulation of system clock, irix_stime on mips and setting the real-time clock.
CAP_SYS_TTY_CONFIG	Defined as 26 in capability.h. Allow configuration of tty devices and vhangup() of tty.
CAP_MKNOD	Defined as 27 in capability.h. Allow the privileged aspects of mknod().
CAP_LEASE	Defined as 28 in capability.h. Allow taking of leases on files.
CAP_AUDIT_WRITE	Defined as 29 in capability.h.
CAP_AUDIT_CONTROL	Defined as 30 in capability.h.
CAP_SETFCAP	Defined as 31 in capability.h. NOT supported on all UNIX OSes as many versions of capability.h stop at 30.
CAP_MAC_OVERRIDE	Defined as 32 in capability.h. Override MAC access. The base kernel enforces no MAC policy. An LSM may enforce a MAC policy, and if it does and it chooses to implement capability based overrides of that policy, this is the capability it should use to do so. NOT supported on all UNIX OSes as many versions of capability.h stop at 30.
CAP_MAC_ADMIN	Defined as 33 in capability.h. Allow MAC configuration or state

[94] For more information see
http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/capability.h
[95] For more information see
http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/capability.h

	changes. The base kernel requires no MAC configuration. An LSM may enforce a MAC policy, and if it does and it chooses to implement capabilitybased checks on modifications to that policy or the data required to maintain it, this is the capability it should use to do so.
<empty string>	This value indicates that no value has been specified and is permitted here to allow for an empty entity which is associated with error and not collected conditions.

2.21 unix-def:inetd_test

The `inetd_test` is used to make assertions about different Internet services associated with a UNIX system, especially information in **`/etc/inet/inetd.conf`** or **`/etc/inetd.conf`**[96]. The `inetd_test` MUST reference one `inetd_object` and zero or more `inetd_states`.


oval-def::TestType
-id : TestIDPattern
-version : unsigned int
-check_existence : ExistenceEnumeration = at_least_one_exists
-check : CheckEnumeration
-state_operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::inetd_test
unix-def::inetd_object
unix-def::inetd_state

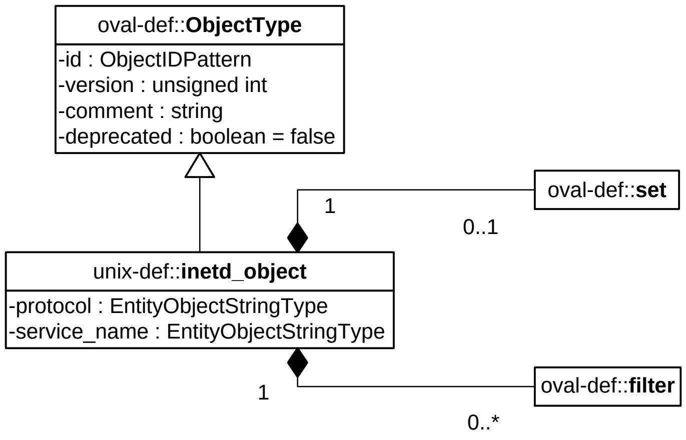
2.21.1 Known Supported Platforms

Some of the latest UNIX platforms are bundled with the xinetd command instead of the inetd command. In this case, the xinetd_test SHOULD be used instead.

2.22 unix-def:inetd_object

The `inetd_object` construct defines the set of Internet services whose associated information should be collected and represented as `inetd_items`[97].

[96] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[97] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf



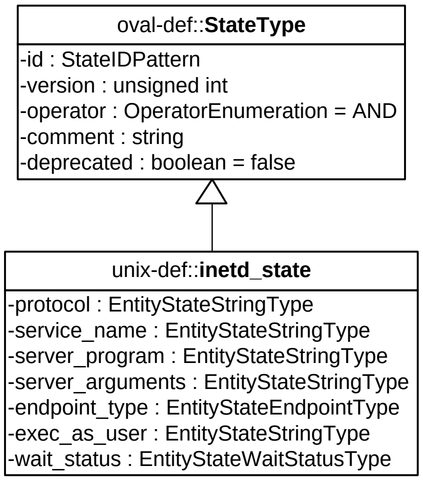
Property	Type	Multiplicity	Nillable	Description
set	oval-def:set	0..1	false	Enables the expression of complex `inetd_objects` that are the result of logically combining and filtering the `inetd_items` that are identified by one or more `inetd_objects`. Please see the OVAL Language Specification for additional information.
protocol	oval-def: EntityObjectStringType	0..1	false	A recognized protocol listed in the file **`/etc/inet/protocols`**, as well as others supported under IPv6. Some of these values in **`/etc/inet/protocols`** include **`tcp`** and **`udp`**[98]. Because **`tcp6`**, **`tcp6only`**, **`udp6`**, and **`udp6only`** are NOT official protocols, they will NOT be listed in the **`/etc/inet/protocols`** file[99]; however, they will still be recognized as **`inetd`** protocol types. The **`inetd`** program uses an **`AF_INET6`** type socket endpoint, which supports BOTH IPv4 and IPv6 client requests.
service_name	oval-def: EntityObjectStringType	0..1	false	The name of a valid service listed in the services file. For RPC services, the value of the service-name field

[98] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=1M&topic=inetd
[99] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf

				consists of the RPC service name or program number, followed by a '/' (slash) and either a version number or a range of version numbers (for example, **`rstatd/2-4`**).
filter	oval-def:filter	0..*	false	Allows for the explicit inclusion or exclusion of `inetd_items` from the set of `inetd_items` collected by an `inetd_object`. Please see the OVAL Language Specification [2] for additional information.

2.23 unix-def:inetd_state

The `inetd_state` construct is used by an `inetd_test` to specify indormation about Internet services on UNIX platforms. This information is located in **`/etc/inet/inetd.conf`** or **`/etc/inetd.conf`**[100].


oval-def::StateType
-id : StateIDPattern
-version : unsigned int
-operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::inetd_state
-protocol : EntityStateStringType
-service_name : EntityStateStringType
-server_program : EntityStateStringType
-server_arguments : EntityStateStringType
-endpoint_type : EntityStateEndpointType
-exec_as_user : EntityStateStringType
-wait_status : EntityStateWaitStatusType

Property	Type	Multiplicity	Nillable	Description
protocol	oval-def:EntityStateStringType	0..1	false	A recognized protocol listed in the file **`/etc/inet/protocols`**, as well as others supported under IPv6. Some of these values in **`/etc/inet/proto`**

[100] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf

				cols include **tcp** and **udp**[101]**.** Because **tcp6**, **tcp6only**, **udp6**, and **udp6only** are NOT official protocols, they will NOT be listed in the **/etc/inet/protocols** file[102]; however, they will still be recognized as **inetd** protocol types. The **inetd** program uses an **AF_INET6** type socket endpoint, which supports BOTH IPv4 and IPv6 client requests.
service_name	oval-def:EntityStateStringType	0..1	false	The name of a valid service listed in the services file. For RPC services, the value of the service-name field consists of the RPC service name or program number, followed by a '/' (slash) and either a version number or a range of version numbers (for example, **rstatd/2-4**).
server_program	oval-def:EntityStateStringType	0..1	false	Either the pathname of a server program to be invoked by inetd to perform the requested service, or the value internal if inetd itself provides the service[103].

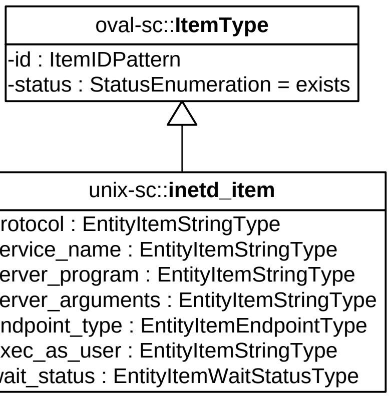
[101] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=1M&topic=inetd
[102] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[103] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf

server_arguments	oval-def:EntityStateStringType	0..1	false	The arguments passed to the server program starting with argv[0][104].
endpoint_type	unix-def: EntityStateEndpointType	0..1	false	The type of socket established by the service for communications[105].
exec_as_user	oval-def:EntityStateStringType	0..1	false	The user name, and optional group name, that the server will run as when it starts up[106].
wait_status	unix-def: EntityStateWaitStatusType	0..1	false	This property takes on the values **`wait`** and **`nowait`**. It specifies whether the server that is invoked by **`inetd`** will take over the listening socket associated with the service, and whether once launched, inetd will wait for that server to exit, if ever, before it resumes listening for new service requests[107].

2.24 unix-sc:inetd_item

The `inetd_item` construct defines the information associated with Internet services on file systems supported by the UNIX platform. This information is located in **`/etc/inet/inetd.conf`** or **`/etc/inetd.conf`**[108].

[104] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[105] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[106] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[107] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[108] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf


oval-sc::ItemType
-id : ItemIDPattern
-status : StatusEnumeration = exists
unix-sc::inetd_item
-protocol : EntityItemStringType
-service_name : EntityItemStringType
-server_program : EntityItemStringType
-server_arguments : EntityItemStringType
-endpoint_type : EntityItemEndpointType
-exec_as_user : EntityItemStringType
-wait_status : EntityItemWaitStatusType

Property	Type	Multiplicity	Nillable	Description
protocol	oval-sc:EntityItemStringType	0..1	false	A recognized protocol listed in the file **`/etc/inet/protocols`**, as well as others supported under IPv6. Some of these values in **`/etc/inet/protocols`** include **`tcp`** and **`udp`**[109]. Because **`tcp6`**, **`tcp6only`**, **`udp6`**, and **`udp6only`** are NOT official protocols, they will NOT be listed in the **`/etc/inet/protocols`** file[110]; however, they will still be recognized as **`inetd`** protocol types. The **`inetd`** program uses an **`AF_INET6`** type socket endpoint, which supports BOTH IPv4 and IPv6 client requests.
service_name	oval-sc:EntityItemStringType	0..1	false	The name of a valid service listed in the services file. For RPC

[109] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=1M&topic=inetd
[110] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf

				services, the value of the service-name field consists of the RPC service name or program number, followed by a '/' (slash) and either a version number or a range of version numbers (for example, **`rstatd/2-4`**).
server_program	oval-sc:EntityItemStringType	0..1	false	Either the pathname of a server program to be invoked by inetd to perform the requested service, or the value internal if inetd itself provides the service[111].
server_arguments	oval-sc:EntityItemStringType	0..1	false	The arguments passed to the server program starting with argv[0][112].
endpoint_type	unix-sc: EntityItemEndpointType	0..1	false	The type of socket established by the service for communications[113].
exec_as_user	oval-sc:EntityItemStringType	0..1	false	The user name, and optional group name, that the server will run as when it starts up[114].
wait_status	unix-sc: EntityItemWaitStatusType	0..1	false	This property takes on the values "wait" and "nowait." It specifies whether the server that is invoked by **`inetd`** will take over the listening socket associated with the service, and whether once launched, **`inetd`** will wait for

[111] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[112] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[113] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[114] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf

				that server to exit, if ever, before it resumes listening for new service requests[115].

2.25 unix-def:EntityStateEndpointType

The `EntityStateEndpointType` defines the values that describe different socket types associated with an Internet service UNIX systems[116].

Enumeration Value	Description
stream	The stream value is used to describe a stream socket.
dgram	The dgram value is used to describe a datagram socket.
raw	The raw value is used to describe a raw socket.
seqpacket	The seqpacket value is used to describe a sequenced packet socket.
tli	The tli value is used to describe all TLI endpoints.
<empty string>	The empty string value is permitted here to allow for empty elements associated with variable references.

2.26 unix-sc:EntityItemEndpointType

The `EntityItemEndpointType` defines the values that describe different socket types associated with an Internet service UNIX systems[117].

Enumeration Value	Description
stream	The stream value is used to describe a stream socket.
dgram	The dgram value is used to describe a datagram socket.
raw	The raw value is used to describe a raw socket.
seqpacket	The seqpacket value is used to describe a sequenced packet socket.
tli	The tli value is used to describe all TLI endpoints.
<empty string>	The empty string value is permitted here to allow for empty elements associated with variable references.

[115] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[116] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[117] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf

2.27 unix-def:EntityStateWaitStatusType

The `EntityStateWaitStatusType` defines the values that describe different wait status types associated with an Internet service UNIX systems[118]. These two types are *'wait'*, and *'nowait'*. It specifies whether the server that is invoked by `inetd` will take over the listening socket associated with the service, and whether once launched, `inetd` will wait for that server to exit, if ever, before it resumes listening for new service requests.

A system administrator SHOULD set the wait-status for datagram servers to *'wait'* and additionally, configure UDP services as *'wait'* instead of *'nowait'*, as it can cause a race condition by which the **`inetd`** program selects on the sockets and the server program reads from the socket. As a result, many server programs will be forked and performance will be severely compromised[119].

Enumeration Value	Description
wait	The server invoked by `inetd` <u>will</u> take over the listening socket associated with the service and once launched, `inetd` <u>will</u> wait for that server to exit, if ever, before it resumes listening for new service requests.
nowait	The server invoked by `inetd` <u>will not</u> take over the listening socket associated with the service and once launched, `inetd` <u>will not</u> wait for that server to exit, if ever, before it resumes listening for new service requests.
<empty string>	The empty string value is permitted here to allow for empty elements associated with variable references.

2.28 unix-sc:EntityItemWaitStatusType

The `EntityItemWaitStatusType` defines the values that describe different wait status types associated with an Internet service UNIX systems[120]. These two types are *'wait'*, and *'nowait'*. It specifies whether the server that is invoked by `inetd` will take over the listening socket associated with the service, and whether once launched, `inetd` will wait for that server to exit, if ever, before it resumes listening for new service requests.

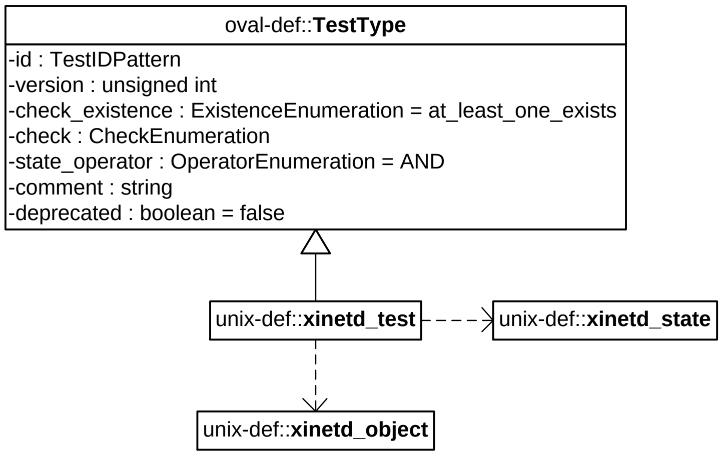
A system administrator SHOULD set the wait-status for datagram servers to *'wait'* and additionally, configure UDP services as *'wait'* instead of *'nowait'*, as it can cause a race condition by which the inetd program selects on the sockets and the server program reads from the socket. As a result, many server programs will be forked and performance will be severely compromised[121].

[118] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[119] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[120] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[121] For more information see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf

Enumeration Value	Description
wait	The server invoked by `inetd` will take over the listening socket associated with the service and once launched, `inetd` will wait for that server to exit, if ever, before it resumes listening for new service requests.
nowait	The server invoked by `inetd` will not take over the listening socket associated with the service and once launched, `inetd` will not wait for that server to exit, if ever, before it resumes listening for new service requests.
<empty string>	The empty string value is permitted here to allow for empty elements associated with variable references.

2.29 unix-def:xinetd_test

The `xinetd_test` is used to make assertions about different Internet services associated with more up-to-date UNIX systems than those covered in the `inetd_test`, especially information in `/etc/xinetd.conf`[122]. The `xinetd_test` MUST reference one `xinetd_object` and zero or more `xinetd_states`.


oval-def::TestType
-id : TestIDPattern
-version : unsigned int
-check_existence : ExistenceEnumeration = at_least_one_exists
-check : CheckEnumeration
-state_operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::xinetd_test
unix-def::xinetd_state
unix-def::xinetd_object

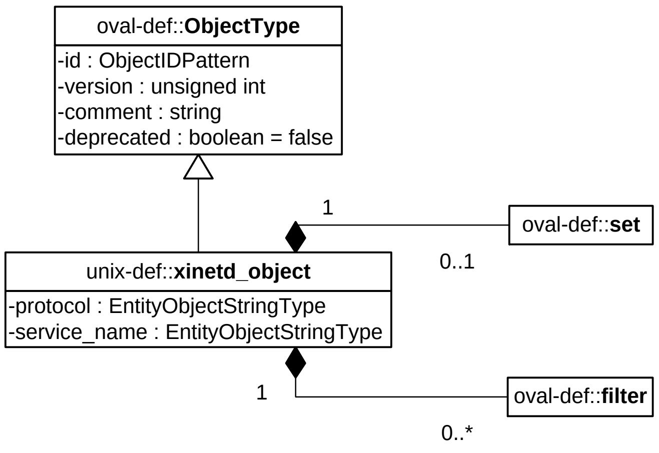
2.29.1 Known Supported Platforms

- Red Hat Enterprise Linux 5
- Mac OSX 10.6
- Solaris 10

[122] For more information see http://linux.die.net/man/5/xinetd.conf

2.30 unix-def:xinetd_object

The `xinetd_object` construct defines the set of Internet services whose associated information should be collected and represented as `xinetd_items`[123].



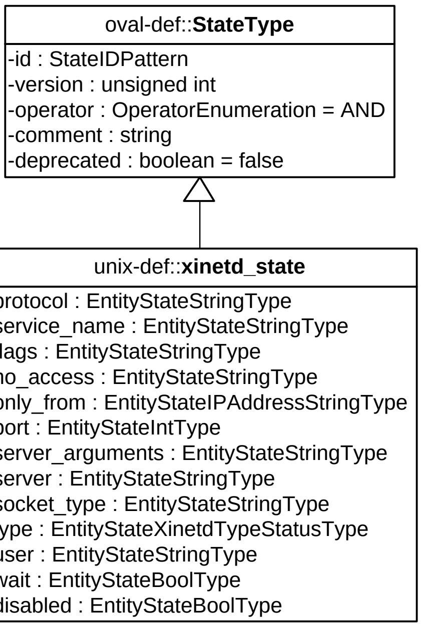
Property	Type	Multiplicity	Nillable	Description
set	oval-def:set	0..1	false	Enables the expression of complex `xinetd_objects` that are the result of logically combining and filtering the `xinetd_items` that are identified by one or more `xinetd_objects`. Please see the OVAL Language Specification[2] for additional information.
protocol	oval-def: EntityObjectStringType	0..1	false	A recognized protocol, such as one listed in the file **`/etc/protocols`**, used by the service. If this property is not defined in the **`xinetd.conf`** file, the default protocol employed by the service will be used[124].
service_name	oval-def: EntityObjectStringType	0..1	false	The name of a valid service listed in the services file[125]. For RPC services, the value of the service-name field consists of the RPC service name or program number, followed by a '/' (slash) and either a version number or

[123] For more information see http://linux.die.net/man/5/xinetd.conf
[124] For more information see http://linux.die.net/man/5/xinetd.conf
[125] For more information see http://linux.die.net/man/5/xinetd.conf

				a range of version numbers (for example, **`rstatd/2-4`**). By default, the service id is the service name.
filter	oval-def:filter	0..*	false	Allows for the explicit inclusion or exclusion of `xinetd_items` from the set of `xinetd_items` collected by an `xinetd_object`. Please see the OVAL Language Specification [2] for additional information.

2.31 unix-def:xinetd_state

The `xinetd_state` construct is used by an `xinetd_test` to specify indormation about Internet services on UNIX platforms. This information is located in **`/etc/xinetd.conf`**[126].


oval-def::StateType
-id : StateIDPattern
-version : unsigned int
-operator : OperatorEnumeration = AND
-comment : string
-deprecated : boolean = false
unix-def::xinetd_state
-protocol : EntityStateStringType
-service_name : EntityStateStringType
-flags : EntityStateStringType
-no_access : EntityStateStringType
-only_from : EntityStateIPAddressStringType
-port : EntityStateIntType
-server_arguments : EntityStateStringType
-server : EntityStateStringType
-socket_type : EntityStateStringType
-type : EntityStateXinetdTypeStatusType
-user : EntityStateStringType
-wait : EntityStateBoolType
-disabled : EntityStateBoolType

Property	Type	Multiplicity	Nillable	Description
protocol	oval-def:EntityStateStringType	0..1	false	A recognized protocol, such as one listed in the file **`/etc/protocols`**, used by the

[126] For more information see http://linux.die.net/man/5/xinetd.conf

				service. If this property is not defined in the **`xinetd.conf`** file, the default protocol employed by the service will be used[127].
service_name	oval-def:EntityStateStringType	0..1	false	The name of a valid service listed in the services file. For RPC services, the value of the service-name field consists of the RPC service name or program number, followed by a '/' (slash) and either a version number or a range of version numbers (for example, **`rstatd/2-4`**).
flags	oval-def:EntityStateStringType	0..1	false	The flags property specifies miscellaneous settings associated with the service. It can take on values such as INTERCEPT, NORETRY, IDONLY, NAMEINARGS, NODELAY, KEEPALIVE, NOLIBWRAP, SENSOR, IPv4, IPv6, LABELLED, and REUSE (deprecated)[128].
no_access	oval-def:EntityStateStringType	0..1	false	Determines the remote hosts to which the particular service is unavailable. Its value can be

[127] For more information see http://linux.die.net/man/5/xinetd.conf
[128] For more information about the different flags see http://linux.die.net/man/5/xinetd.conf

				specified in the same way as the value of the **only_from** property. These two properties determine the access control enforced by **xinetd**. If none of the two is specified for a service, the service is available to anyone.
only_from	oval-def: EntityStateIPAddressStringType	0..1	false	Determines the remote hosts to which the particular service is available. Its value is a list of IP addresses which can be specified in any combination of a numerical address, a factorized address, a network name, a host name, and/or an ip address/netmask range[129].
port	oval-def:EntityStateIntType	0..1	false	Determines the service port. If this property is specified for a service listed in **/etc/services**, it SHOULD be equal to the port number listed in that file.
server	oval-def:EntityStateStringType	0..1	false	Determines the program to execute for this service.
server_arguments	oval-def:EntityStateStringType	0..1	false	Determines the arguments passed to the server. Unlike **inetd**, the server name SHOULD NOT be included[130].

[129] For more information about the specific host formatting available see http://linux.die.net/man/5/xinetd.conf
[130] For more information see http://linux.die.net/man/5/xinetd.conf

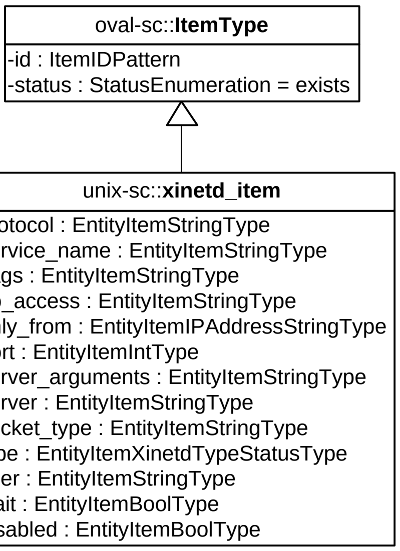
socket_type	oval-def:EntityStateStringType	0..1	false	Specifies the type of socket that is used by the service[131].
type	unix-def: EntityStateXinetdTypeStatusType	0..1	false	Specifies the type of the service. Any combination of the values RPC, INTERNAL, TCPMUX/TCPMUXPLUS, or UNLISTED can be used[132].
user	oval-def:EntityStateStringType	0..1	false	Determines the uid for the server process. The user property can either be numeric or a name (recommended). If a name is given the user name must exist in **`/etc/passwd`**. This attribute is ineffective if the effective user ID of **`xinetd`** is NOT super-user[133].
wait	oval-def:EntityStateBoolType	0..1	false	This property determines if the process is single or multi-threaded and whether or not **`xinetd`** accepts the connection or the server program accepts the connection[134].
disabled	oval-def:EntityStateBoolType	0..1	false	A property of which when set to *true*, the service is disabled and not starting, and

[131] For more information see http://linux.die.net/man/5/xinetd.conf
[132] For more information see http://linux.die.net/man/5/xinetd.conf
[133] For more information see http://linux.die.net/man/5/xinetd.conf
[134] For more information about the implications of a single or multi-threaded service, see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf

				when set to *false*, the service is enabled[135].

2.32 unix-sc:xinetd_item

The `xinetd_item` construct defines the information associated with Internet services on file systems supported by the UNIX platform. This information is located in **`/etc/xinetd.conf`**[136].


oval-sc::**ItemType**
-id : ItemIDPattern
-status : StatusEnumeration = exists
unix-sc::**xinetd_item**
-protocol : EntityItemStringType
-service_name : EntityItemStringType
-flags : EntityItemStringType
-no_access : EntityItemStringType
-only_from : EntityItemIPAddressStringType
-port : EntityItemIntType
-server_arguments : EntityItemStringType
-server : EntityItemStringType
-socket_type : EntityItemStringType
-type : EntityItemXinetdTypeStatusType
-user : EntityItemStringType
-wait : EntityItemBoolType
-disabled : EntityItemBoolType

Property	Type	Multiplicity	Nillable	Description
protocol	oval-sc:EntityItemStringType	0..1	false	A recognized protocol, such as one listed in the file **`/etc/protocols`**, used by the service. If this property is not defined in the **`xinetd.conf`** file, the default protocol employed by the service will be used[137].
service_name	oval- sc:EntityItemStringType	0..1	false	The name of a valid service listed in the services file. For RPC services, the value of the service-name field consists of the RPC

[135] For more information about the implications of a single or multi-threaded service, see http://cims.nyu.edu/cgi-systems/man.cgi?section=4&topic=inetd.conf
[136] For more information see http://linux.die.net/man/5/xinetd.conf
[137] For more information see http://linux.die.net/man/5/xinetd.conf

				service name or program number, followed by a '/' (slash) and either a version number or a range of version numbers (for example, **`rstatd/2-4`**).
flags	oval- sc:EntityItemStringType	0..*	false	The flags property specifies miscellaneous settings associated with the service. It can take on values such as INTERCEPT, NORETRY, IDONLY, NAMEINARGS, NODELAY, KEEPALIVE, NOLIBWRAP, SENSOR, IPv4, IPv6, LABELLED, and REUSE (deprecated)[138].
no_access	oval- sc:EntityItemStringType	0..*	false	Determines the remote hosts to which the particular service is unavailable. Its value can be specified in the same way as the value of the **`only_from`** property. These two properties determine the access control enforced by **`xinetd`**. If none of the two is specified for a service, the service is available to anyone.
only_from	oval- sc: EntityItemIPAddressStringType	0..*	false	Determines the remote hosts to which the particular service is available. Its value is a list of IP addresses which can be specified in any combination of

[138] For more information about the different flags see http://linux.die.net/man/5/xinetd.conf

				a numerical address, a factorized address, a network name, a host name, and/or an ip address/netmask range[139].
port	oval- sc:EntityItemIntType	0..1	false	Determines the service port. If this property is specified for a service listed in **`/etc/services`**, it SHOULD be equal to the port number listed in that file.
server	oval- sc:EntityItemStringType	0..1	false	Determines the program to execute for this service.
server_arguments	oval- sc:EntityItemStringType	0..1	false	Determines the arguments passed to the server. Unlike **`inetd`**, the server name SHOULD NOT be included[140].
socket_type	oval- sc:EntityItemStringType	0..1	false	Specifies the type of socket that is used by the service[141].
type	unix-sc: EntityItemXinetdTypeStatusType	0..1	false	Specifies the type of the service[142].
user	oval- sc:EntityItemStringType	0..1	false	Determines the uid for the server process. The user attribute can either be numeric or a name (recommended). If a name is given the user name must exist in **`/etc/passwd`**. This attribute is ineffective if the effective user ID of `xinetd` is NOT super-user[143].

[139] For more information about the specific host formatting available see http://linux.die.net/man/5/xinetd.conf
[140] For more information see http://linux.die.net/man/5/xinetd.conf
[141] For more information see http://linux.die.net/man/5/xinetd.conf
[142] For more information see http://linux.die.net/man/5/xinetd.conf
[143] For more information see http://linux.die.net/man/5/xinetd.conf

wait	oval- sc:EntityItemBoolType	0..1	false	This attribute determines if the process is single or multi-threaded and whether or not xinetd accepts the connection or the server program accepts the connection[144].
disabled	oval- sc:EntityItemBoolType	0..1	false	A property of which when set to *true*, the service is disabled and not starting, and when set to *false*, the service is enabled[145].

2.33 unix-def:EntityStateXinetdTypeStatusType

The `EntityStateXinetdTypeStatusType` defines the values that describe the different types of Internet service functionality on UNIX systems[146].

Enumeration Value	Description
INTERNAL	The INTERNAL type is used to describe services like echo, chargen, and others whose functionality is supplied by xinetd itself.
RPC	The RPC type is used to describe services that use remote procedure call ala NFS.
UNLISTED	The UNLISTED type is used to describe services that aren't listed in /etc/protocols or /etc/rpc.
TCPMUX	The TCPMUX type is used to describe services that conform to RFC 1078. This type indiciates that the service is responsible for handling the protocol handshake.
TCPMUXPLUS	The TCPMUXPLUS type is used to describe services that conform to RFC 1078. This type indicates that xinetd is responsible for handling the protocol handshake.
<empty string>	The empty string value is permitted here to allow for empty elements associated with variable references.

[144] For more information about the implications of a single or multi-threaded service, see http://linux.die.net/man/5/xinetd.conf
[145] For more information about the implications of a single or multi-threaded service, see http://linux.die.net/man/5/xinetd.conf
[146] For more information see http://linux.die.net/man/5/xinetd.conf

2.34 unix-sc:EntityItemXinetdTypeStatusType

The `EntityItemXinetdTypeStatusType` defines the values that describe the different types of Internet service functionality on UNIX systems[147].

Enumeration Value	Description
INTERNAL	The INTERNAL type is used to describe services like echo, chargen, and others whose functionality is supplied by xinetd itself.
RPC	The RPC type is used to describe services that use remote procedure call ala NFS.
UNLISTED	The UNLISTED type is used to describe services that aren't listed in /etc/protocols or /etc/rpc.
TCPMUX	The TCPMUX type is used to describe services that conform to RFC 1078. This type indiciates that the service is responsible for handling the protocol handshake.
TCPMUXPLUS	The TCPMUXPLUS type is used to describe services that conform to RFC 1078. This type indicates that xinetd is responsible for handling the protocol handshake.
<empty string>	The empty string value is permitted here to allow for empty elements associated with variable references.

[147] For more information see http://linux.die.net/man/5/xinetd.conf

Appendix A – Normative References

[1] RFC 2119 – Key words for use in RFCs to Indicate Requirement Levels
http://www.ietf.org/rfc/rfc2119.txt

[2] The OVAL Language Specification
http://oval.mitre.org/language/version5.10#specification

Appendix B - Change Log

Version 5.11 Revision 5 – December 18, 2014

- Updated version and date information for the Official 5.11 Release.

Version 5.11 Revision 4 – December 01, 2014

- Updated version and date information for 5.11 Release Candidate 2.

Version 5.11 Revision 3 – November 18, 2014

- Updated version and date information for 5.11 Release Candidate 1.

Version 5.11 Revision 2 – September 25, 2013

- **No changes were made other than updating the document version information.**

Version 5.11 Revision 1 – February 20, 2013

- Added documentation clarifying the expected behavior for the has_extended_acl entity in the unix-sc:file_item. This addresses https://github.com/OVALProject/Language/issues/12.
- Updated version and date information for 5.11 Draft 1.

Version 5.10 Revision 1 – April 4, 2012

- Published initial revision of the version 5.10.1 UNIX extension specification.

Appendix C – Terms and Acronyms