uSched Services and Tools

Reference Manual - Rev. A01

7th March 2015

Contents

List of Tables

Part I
Overview

1 Introduction

uSched Services and Tools (or simply uSched) provide an interface to schedule commands to be executed at a particular time, optionally repeating them over a specificied interval, and optionally stopping them at any other particular time. It provides a simple and intuitive structured language that is intepreted via a command line client, but can also be integrated into any programming language through its client libraries and bindings. It also operates as a client/server, where requests performed by clients can affect local or remote machines where uSched services are running.

2 Components

uSched architecture is divided into five (5) main components: The uSched Admin, uSched Client, uSched Daemon, uSched Executer and uSched Monitor. There are only two components that require direct user interaction: uSched Admin (usa) for configuration and uSched Client (usc) for scheduling requests. All the other components are self-managed and shall not be called directly.

2.1 uSched Admin (usa)

The uSched Admin component is a command line binary utility named 'usa'. Its purpose is to manage and handle configuration parameters that affect the core, network, authentication and users used by the other uSched components. Changes performed using uSched Admin won't be effective until a COMMIT operation is performed. Also any uncommitted changes can be rolled back by issuing the ROLLBACK operation. This utility requires super-user privileges to be used.

2.2 uSched Client (usc)

The uSched Client component is a command line binary utility named 'usc'. It allows the users to perform scheduling requests to the uSched Daemon component. This is the only component that can be executed by any non-blacklisted (or all white-listed) users. With a default installation, all the other uSched components will require super-user privileges to be used.

2.3 uSched Daemon (usd)

The uSched Daemon component is a system service that receives and processes all the scheduling requests performed by uSched clients. It manages the authentication and authorization, schedulers and is the only component that directly communicates with the uSched Executer. With a default installation, this component requires super-user privileges to be started, although it will drop it's privileges after start-up routines are performed.

2.4 uSched Executer (use)

The uSched Executer component is a system service that receives execution requests from the uSched Daemon. Every time a scheduled entry is triggered by the uSched Daemon scheduler, it will be passed to the uSched Executer to be executed with the required privileges. This component requires super-user privileges with a default installation in order to be able to execute scheduled entries for all the users.

2.5 uSched Monitor (usm)

The uSched Monitor component is a simple command line binary utility that starts and daemonizes the uSched Daemon and uSched Executer and monitors their processes state. It will restart or reload the services whenever it is required to do so. This utility requires super-user privileges to operate correctly.

3 Portability

uSched is designed to be compliant with any POSIX operating system. There are some features that may not be enabled by default in the case that some non-portable calls being unavailable for the target operating system, such as chroot(). To disable such calls and features, set the CONFIG_POSIX_FULL definition to 1 in the include/config.h file. uSched Client will also compile on most Windows versions and the API bindings for C# are promptly available on the project bindings/csharp directory.

Part II
Installation

4 Getting uSched

Currently, uSched Services and Tools can be downloaded from the following GitHub URL:

https://github.com/ucodev/usched

To clone the repository performed the following command (GIT must be installed on the system):

```
$ git clone https://github.com/ucodev/usched
```

It can also be downloaded as a ZIP package via 'wget' utility:

```
$ wget https://github.com/ucodev/usched/archive/master.zip
$ unzip -d usched master.zip
$ cd usched
```

Or via 'curl' utility:

```
$ curl -o https://github.com/ucodev/usched/archive/master.zip
$ unzip -d usched master.zip
$ cd usched
```

5 Generic Setup

After downloading (and extracting) the uSched source package, it is recommended to read the version specific README.txt and INSTALL.txt files that are present on the doc/ direcotry. Typically, it can be built and installed by executing the following single command from the source base directory, as superuser:

```
# ./deploy
```

This will install all the required dependencies and uSched base system daemons and tools. For System-Specific initialization scripts and utilities, please refer to the next section (6).

6 System-Specific Procedures

Since initialization scripts differ on many systems, they currently need to be installed through a separe procedure that isn't included in the 'deploy' script. The generic usage to install System-Specific utilieies is:

```
# make install_<system>
```

Where <system> is a keyword that identifies the target system. The supported keywords are described in the following sub-sections.

6.1 Linux/Debian

The available keywords are:

```
classic_debian
systemd
```

If the Debian version doesn't fully support systemd by default, the classic_debian keyword shall be used:

```
# make install_classic_debian
```

Otherwise, for Debian versions with systemd enabled:

```
# make install_systemd
```

6.2 Linux/RedHat

The available keywords are:

```
classic_redhat
systemd
```

If the RedHat version doesn't fully support systemd by default, the classic_redhat keyword shall be used:

```
# make install_classic_redhat
```

Otherwise, for RedHat versions with systemd enabled:

```
# make install_systemd
```

6.3 BSD/FreeBSD

The available keywords are:

```
freebsd
```

To install the FreeBSD initialization scripts, execute:

```
# make install_freebsd
```

Part III
Usage

7 uSched Client

The uSched Client component is the interface between the users and the uSched Daemon. It allows the user to install, remove or show Scheduled Entries.

7.1 Generic Usage

The generic usage for uSched Client is:

> usc [OPTIONS] OP SUBJ { PREP [AVERB ARG | ARG ADVERB] [CONJ ...] }

Usage examples:

> $ usc run 'df -h > /tmp/output' in 10 seconds
> $ usc run 'take_screenshot.sh' on hour 8 then every 30 minutes
> $ usc run 'do_work.py' on hour 8 then every 1 hour until to hour 17
> $ usc run 'do_weekend_stuff.pl' on weekday saturday then every week
> # usc run 'tcpdump -i eth0 > /tmp/tcpdump.log' now and 'kill -TERM `pidof tcpdump`' in 1 hour
> $ usc show all

7.2 Options

The available command lines options are:

> **-h** Displays help
> **-H** IP Address of the remote server
> **-p** TCP port of the remote server
> **-U** Username for remote authentication
> **-P** Password for remote authentication

Example:

> $ usc -H 192.168.0.1 -U testuser -P testpass show all

7.3 Operations

uSched Client Operations are the first mandaory argument. The Client expects at least two (2) mandatory arguments to be accepted for the Operations SHOW and STOP (the Operation itself and a Subject), and three (3) mandatory arguments for the RUN Operation (the Operation itself, a Subject and a Preposition): The following sub-sections describe in more detail the usage for the currently supported Operations..

7.3.1 RUN

The RUN Operation translates and requests the installation of one or more Scheduler Entries on uSched Services..A Scheduler Entry structure is described by the following table:

Field	Type	Description
ID	64-bit UINT	Scheduler Entry ID
Username	String	Username, if remotely requested
UID	64-bit UINT	Command must be executed as UID
GID	64-bit UINT	Command must be executed as GID
Trigger	64-bit UINT	Next execution of entry (UNIX Timestamp)
Step	64-bit UINT	Incrementation value for Trigger
Expire	64-bit UINT	Remove Entry after this UNIX Timestamp
Command	String	Shell command (Subject)

Table 1: Scheduler Entry Structure

Generic RUN Usage:

run SUBJ PREP [ADVERB ARG | ARG ADVERB] [CONJ ...]

Usage examples:

run 'ntpdate' now
run 'ntpdate' in 1 hour
run 'ntpdate' on hour 22
run 'ntpdate' now then every 6 hours
run 'ntpdate' on weekday monday then every 1 week
run 'ntpdate' on day 15 then every 1 month until datetime is '2018/01/01 01:00:00'
run 'ntpdate' now then every 3 hours and 'date >> /tmp/date.log' in 3 hours then every 3 hours
run 'tcpdump -i eth1 > /tmp/tcpdump_eth1.log' in 5 minutes and 'kill -TERM `pidof tcpdump`' in 10 minutes
run 'free >> /tmp/mem_info.log' now then every 5 seconds while in 10 minutes

After each successfully installed request, the uSched Client will return the corresponding Entry ID for each of the installed Scheduled Entries. To retrieve information regarding the currently installed Scheduled Entries, please refer to section **7.3.3 SHOW**.

7.3.2 STOP

The STOP operation cancels and removes a previously installed Scheduled Entry. The generic STOP Usage is:

```
stop SUBJ
```

Where SUBJ (Subject) must be at least a valid and installed Entry ID, or the special SUBJ value 'all', which will stop all the currently installed Scheduled Entries for the calling user. Multiple Entry IDs can be specified in a comma separated list, as specified in the following usage examples:

```
stop 0xD9947D5E546B3707
stop 0xD9947D5E546B3707,0x0B30462B97160FC4
stop 0xD9947D5E546B3707,0x0B30462B97160FC4,0xEEAD3BBAC6BBF642
stop all
```

Note that although in the examples above the hexadecimal format was used, the Entry ID value can be also specified in decimal format.

7.3.3 SHOW

The SHOW operation displays the currently installed Scheduled Entries. It's generic Usage is:

```
show SUBJ
```

Where SUBJ (Subject) must be at least a valid and installed Entry ID, or the special SUBJ value 'all', which will display all the currently installed entries for the calling user. Multiple Entry IDs can be specified in a comma separated list, as specified in the following usage examples:

```
show 0xD9947D5E546B3707
show 0xD9947D5E546B3707,0x0B30462B97160FC4
show 0xD9947D5E546B3707,0x0B30462B97160FC4,0xEEAD3BBAC6BBF642
show all
```

Note that although in the examples above the haxadecimal format was used, the Entry ID value can be also specified in decimal format.

7.4 Subjects

A Subject is the second mandatory argument of the uSched Client. Its meaning is directly related to the Operation that preceeds it. The following subsections describe the acceptable subjects for each uSched Client operation.

7.4.1 Subjects of RUN Operation

The RUN Operation Subject is any valid shell (/bin/sh) command. Each subject is passed to the -c option of /bin/sh. There are several uSched specific variables that can be used in the Subject string that are replaced each time the Entry ID is triggered to be executed by the uSched Daemon scheduler. These variables are described in the table below:

Variable	Description
@@id@@	Current Entry ID
@@username@@	Username used in the remote authentication for the current Entry
@@uid@@	The UID of the current Entry
@@gid@@	The GID of the current Entry
@@trigger@@	The current trigger value of the Entry (UNIX timestamp)
@@step@@	The current step value of the Entry (in seconds)
@@expire@@	The expire value of the Entry (UNIX timestamp)

Table 2: RUN Subject Variables

7.4.2 Subjects of SHOW Operation

The SHOW Operation Subject is any valid Entry ID or a comma separated list of Entry IDs present on the uSched Daemon scheduling list, or the special subject 'all' that targets all the Entry IDs.

7.4.3 Subjects of STOP Operation

The STOP Operation Subject is any valid Entry ID or a comma separated list of Entry IDs present on the uSched Daemon scheduling list, or the special subject 'all' that targets all the Entry IDs.

7.5 Prepositions

Prepositions are the third mandatory argument of the uSched Client and the acceptable keywords after a Subject and/or after a Conjunction. Valid Prepositions after a Subject are the IN, NOW and ON, while the valid Prepositions after a Conjunction are the EVERY and TO.

7.5.1 EVERY

Indicates that the following adverbial of time value is the step of the scheduled entry. The following example will execute the command 'df -h >> /tmp/out.txt' now, then every 5 seconds:

```
$ usc run 'df -h >> /tmp/out.txt' now then every 5 seconds
```

EVERY Preposition always take the adverbial of time format as ARG ADVERB (as in, <value> <unit>).

7.5.2 IN

If used after a Subject, it indicates that the entry should be first triggered in a specified amount of time, defined by the value of the following adverbial of time. If used after a WHILE conjunction, it indicates that the entry should be stopped in a specified amount of time, defined by the value of the following adverbial of time. The followig example will execute the command 'ntpdate' in 10 minutes:

```
$ usc run ntpdate in 10 minutes
```

IN Preposition always take the adverbial of time format as ARG ADVERB (as in, <value> <unit>).

7.5.3 NOW

Indicates that the entry should be first triggered right after it is installed. The following example will execute the command 'take_shot.sh' right after the entry is installed:

```
$ usc run take_shot.sh now
```

7.5.4 ON

Indicates that the entry should be first triggered on a specified point in time, defined by the value of the following adverbial of time. The following command will execute the command 'do_backups.sh' on day 15 of the month March of the year 2015:

```
$ usc run do_backups.sh on date 2015-03-15
```

ON Preposition always take the adverbial of time format as ADVERG ARG (as in, <unit> <value>).

7.5.5 TO

Only acceptable after an UNTIL conjuction, indicating that the entry should be stopped on a specified point in time, defined by the value of the following adverbial of time. The following command will execute the command ntpdate now, then every 6 hours, and stop the entry when the month is May:

```
$ usc run ntpdate now then every 6 hours until to month 5
```

TO Preposition always take the adverbial of time format as ADVERG ARG (as in, <unit> <value>).

7.6 Adverbials of Time

The Adverbials of Time defines a time value based on the specified unit. Units ranges from seconds to years and also may take the form of special values such as weekdays, dates, datetimes and timestamps.

7.6.1 SECONDS

Define a time value in seconds as a positive integer (ARG ADVERB), or a specific second.ranging from 0 to 59 (ADVERB ARG).

```
$ usc run sync_routine.sh now then every 30 seconds
```

7.6.2 MINUTES

Define a time value in minutes as a positive integer (ARG ADVERB), or a specific minute ranging from 0 to 59 (ADVERB ARG).

```
$ usc run sync_routine.sh on minute 10 then every 10 minutes
```

7.6.3 HOURS

Define a time value in hours as a positive integer (ARG ADVERB), or a specific hour ranging from 0 to 23 (ADVERB ARG).

```
$ usc run sync_routine.sh on hour 6 then every 24 hours
```

7.6.4 DAYS

Define a time value in days as a positive integer (ARG ADVERB), or a specific day ranging from 1 to 31 (ADVERB ARG).

```
$ usc run sync_routine.sh on day 11 then every 2 days
```

7.6.5 WEEKS

Define a time value in weeks as a positive integer (ARG ADVERB), or a specific week ranging from 1 to 52 (ADVERB ARG).

```
$ usc run sync_routine.sh on week 32 then every 2 weeks
```

7.6.6 MONTHS

Define a time value in months as a positive integer (ARG ADVERB), or a specific month ranging from 1 to 12 (ADVERB ARG).

```
$ usc run sync_routine.sh on month 4 then every 1 month
```

7.6.7 YEARS

Define a time value in years as a positive integer (ARG ADVERB), or a specific year ranging from 0 to 2100 (ADVERB ARG).

```
$ usc run sync_routine.sh on year 2016 then every 1 year
```

7.6.8 WEEKDAYS

Define a specific weekday ranging from 0 to 6 (ADVERB ARG), or the special keywords sunday, monday, thursday, wednesday, tuesday, friday and saturday.

```
$ usc run sync_routine.sh on weekday monday then every 2
weeks
```

7.6.9 TIME

Define a specific time in the format HH:MM:SS (ADVERB ARG).

```
$ usc run sync_routine.sh on time 18:00:00 then every 10 minutes
```

7.6.10 DATE

Define a specific date in the format YYYY-MM-DD (ADVERB ARG).

```
$ usc run sync_routine.sh on date 2015/03/15 then every 15 days
```

7.6.11 DATETIME

Define a specific datetime in the format YYYY-MM-DD HH:MM:SS (ADVERB ARG).

```
$ usc run sync_routine.sh on datetime '2015/03/15 18:00:00'
then every 10 minutes until to datetime '2015/04/30 10:00:00'
```

7.6.12 TIMESTAMP

Define a specific timestamp. The value must be a positive integer (ADVERB ARG). Timestamp values lower than the current timestamp are automatically replaced with the current timestamp.

```
$ usc run sync_routine.sh on timestamp 1425544000 then every
3600 seconds until to timestamp 1425644000
```

7.7 Conjunctions

Conjunctions are used to aggregate prepositions. There is no limit for the number of conjunctions that can be used in a single uSched Client command. The expected keyword after a conjunction is always a preposition.

7.7.1 AND

The AND Conjunction indicates that the next Preposition specifies another scheduling entry for the current subject. The following command will execute the 'sync' binary now and on month May:

```
$ usc run 'sync' now and on month 5
```

7.7.2 THEN

The THEN Conjunction expects an EVERY Preposition, being this the only valid preposition after this Conjunction. The following command will execute the 'sync' binary every 5 seconds:

```
$ usc run sync now then every 5 seconds
```

7.7.3 UNTIL

The UNTIL Conjunction expects a TO Preposition, being this the only valid prepostion after this Conjunction. The UNTIL TO statement indicates that the scheduling entry will be stopped after this point in time is reached. The following command will execute the 'sync' binary every 5 seconds until to time is '18:00:00':

```
$ usc run sync now then every 5 seconds until to time 18:00:00
```

7.7.4 WHILE

The WHILE Conjunction expects a IN Preposition, being this the only valid preposition after this Conjunction. The WHILE IN statement indicates that the scheduling entry will be executed during the specified value as its adverbial of time, and stopped after it expires. For example, the following command will execute the binary 'sync' every 5 seconds while in the next 30 seconds (as in, it will be executed 6 times):

```
$ usc run sync now then every 5 seconds while in 30 seconds
```

8 uSched Admin

The uSched Admin component is one of the interfaces between the system administrator and the uSched Daemon and uSched Executer. It allows the administrator to configure the daemon.

8.1 Generic Usage

The generic usage for uSched Admin is:

```
usa OP CATEGORY [ ARG1 ARG2 ... ]
usa OP CATEGORY [ COMPONENT PROPERTY VALUE ]
```

Usage examples:

```
$ usa change auth remote users 1
$ usa commit auth
$ usa add users testuser 10000 10000
$ usa show users
$ usa rollback users
$ usa show all
$ usa commit all
```

8.2 Operations

The uSched Admin Operations affect the current configuration of the uSched Daemon and uSched Executer. Each Operation will mangle the target category accordingly.

8.2.1 ADD

Adds a new Component Property to the target Category. This operation is only valid on AUTH and USERS Categories. Usage example:

```
# usa add users testuser 10000 10000
Password:
# usa commit users
```

8.2.2 CHANGE

Changes a Component Property on the target Category. This operation is valid for all Categories, except the ALL category. Usage example:

```
# usa change auth remote users 1
# usa commit auth
# usa change users testuser 5000 5000
Password:
# usa commit users
```

8.2.3 COMMIT

Commits the current Component Properties to the target Category. This operation is valid for all Categories. If the ALL Category is used, all the Categories are affected. Usage example:

```
# usa commit all
# usa commit core
```

8.2.4 DELETE

Deletes a Property from the target Category. This operation is only valid on AUTH and USERS Categories. Usage example:

```
# usa delete users testuser
# usa commit users
```

8.2.5 ROLLBACK

Rollback the uncommited changes from the target Category. This operation is valid for all Categories. If the ALL Category is used, all the Categories are affected. Usage example:

```
# usa rollback users
# usa rollback all
```

8.2.6 SHOW

Shows the Component Properties of the target Categoy. This operation is valid for all Categories. If the ALL Category is used, all the Categories are shown. Usage example:

```
# usa show core
# usa show all
# usa show network
# usa show auth
# usa show core
# usa show users
```

8.3 Categories

There are four (4) main configuration Categories available: The AUTH, CORE, NETWORK and USERS. There is a fifth special category named ALL that do not implement Components nor Properties, being its purpose to affect all the four main categories at once.

8.3.1 ALL

The ALL Category doesn't contain Components nor Properties. It should only be used on Operations that are intended to affect all the main Categories.

8.3.2 AUTH

The AUTH Category configures the authentication and authorization mechanisms of the uSched Daemon and Executer. It contains the BLACKLIST, LOCAL, REMOTE and WHITELIST Components.

8.3.3 CORE

The CORE Category configures the core parameters of the uSched Daemon and Executer. It contains the DELTA, JAIL, PMQ, PRIVDROP, SERIALIZE and THREAD Components.

8.3.4 NETWORK

The NETWORK Category configures the network parameters of the uSched Daemon. It contains the BIND, CONN and SOCK Components.

8.3.5 USERS

The USERS Category configures the authorized remote users that connect to the uSched Daemon. This Category does not implement any Components and uses a special syntax for the ADD, CHANGE and DELETE Operations:

```
# usa add users USERNAME UID GID
# usa change users USERNAME UID GID
# usa delete users USERNAME
```

8.4 Components

The Categories CORE, NETWORK and AUTH contain a set of configurable Components. The following subsections describe all the available Components respective Category from which they belong.

8.4.1 BIND

Available on NETWORK Category. Contains the Properties ADDR and PORT. Used to configure the binding address and port of the uSched Daemon TCP service. Usage examples:

```
# usa change network bind addr 127.0.0.1
# usa change network bind port 7600
# usa commit network
```

8.4.2 BLACKLIST

Available on AUTH Category. Contains the Properties UID and GID. Used to disallow specific GIDs and UIDs to authenticate on the uSched Daemon. Blacklists take precedence over whitelists. Usage examples:

```
# usa change auth blacklist uid 10000,20000
# usa add auth blacklist uid 30000
# usa delete auth blacklist uid 20000
# usa commit auth
# usa change auth blacklist uid ”
# usa commit auth
```

8.4.3 CONN

Available on NETWORK Category. Contains the Properties LIMIT and TIMEOUT. Used to specify the connection controls of the uSched Daemon. Usage examples:

```
# usa change network conn addr 0.0.0.0
# usa change network conn port 7600
# usa commit network
```

8.4.4 DELTA

Available on CORE Category. Contains the Properties NOEXEC and RELOAD. Used to control the no execution and reload tolerances of the uSched Daemon. Usage examples:

```
# usa change core delta noexec 5
# usa change core delta reload 900
# usa commit core
```

8.4.5 JAIL

Available on CORE Category. Contains the Property DIR. Used to specify a jail directory that the uSched Daemon will chroot() into. Usage examples:

```
# usa change core jail dir /var/cache/usched/jail
# usa commit core
```

8.4.6 LOCAL

Available on AUTH Category. Contains the Property USE. Used to enable or disable the access of the local system users to the uSched Daemon. Usage examples:

```
# usa change auth local use 1
# usa commit auth
```

8.4.7 PMQ

Available on CORE Category. Contains the Properties MSGMAX, MSGSIZE and NAME. Used to configure the Posix Message Queue that allows the communication between the uSched Daemon and uSched Executer. Usage examples:

```
# usa change core pmq msgmax 128
# usa change core pmq msgsize 1024
# usa change core name /uschedpmq01
# usa commit core
```

8.4.8 PRIVDROP

Available on CORE Category. Contains the Properties GROUP and USER. Used to indicate an unprivileged user and group that uSched Daemon will drop privileges to. Usage examples:

```
# usa change core privdrop group nogroup
# usa change core privdrop user nobody
# usa commit core
```

8.4.9 REMOTE

Available on AUTH Category. Contains the Property USERS. Used to enable or disable the remote access to the uSched Daemon. Usage example:

```
# usa change auth remote users 1
# usa commit auth
```

8.4.10 SERIALIZE

Available on CORE Category. Contains the Property FILE. Used to speficy the serialization file used by the uSched Daemon to serialize states. Usage example:

```
# usa change core serialize file /var/cache/usched/daemon.dat
# usa commit core
```

8.4.11 SOCK

Available on NETWORK Category. Contains the Property NAME. Used to define the named socket, used by uSched Client and uSched Daemon. Usage example:

```
# usa change network sock name /var/run/usched.sock
# usa commit network
```

8.4.12 THREAD

Available on CORE Category. Contains the Properties PRIORITY and WORKERS. Used to control the priority and number of uSched Daemon workers. Usage examples:

```
# usa change core thread priority 20
# usa change core thread workers 5
# usa commit core
```

8.4.13 WHITELIST

Available on AUTH Category. Contains the Properties GID and UID. Used to allow specific GIDs and UIDs to authenticate on the uSched Daemon. Usage examples:

```
# usa change auth whitelist uid 10000,20000
# usa add auth whitelist uid 30000
# usa delete auth whitelist uid 20000
# usa commit auth
# usa change auth whitelist uid ”
# usa commit auth
```

8.5 Properties

The following subsections describe the available Component Properties.

8.5.1 ADDR

An IPv4 address.

8.5.2 DIR

A valid directory. This is a string value.

8.5.3 FILE

A valid, or possible to be created, filename. This is a string value.

8.5.4 GID

A Group ID. This is an integer value.

8.5.5 GROUP

A system group name. This is a string value.and must be a valid system group name.

8.5.6 LIMIT

The upper limit of a Component. This is a value greater than 0.

8.5.7 MSGMAX

The maximum number of messages a Component can hold. This is an integer value greater than 0.

8.5.8 MSGSIZE

The maximum size of the messages belonging to a Component. This is an integer value greater than 0.

8.5.9 NAME

The name of a component. This is a string value.

8.5.10 NOEXEC

The number of seconds after which a Component should ignore executions. This is an integer value greater than 0.

8.5.11 PORT

The port number of a Component. This is an integer ranging from 1 to 65535.

8.5.12 PRIORITY

The priority of a component. This is an integer value ranging from 0 (highest priority) and 30 (lowest priority).

8.5.13 RELOAD

The number of elapsed seconds that trigger a component to be reloaded. This is an integer value.

8.5.14 TIMEOUT

The number of seconds of inactivity to consider a component as expired. This is an integer value.

8.5.15 UID

A User ID. This is an integer value.

8.5.16 USE

Whether a Component shall be used. This is a boolean value.

8.5.17 USER

A system user name. This is a string value and must be a valid system user name.

8.5.18 USERS

The availability of users for a Component. This is a boolean value.

8.5.19 WORKERS

The maximum number of workers that a Component will use. This is a positive, greater than zero, integer.

Part IV
Administration and Configuration

9 uSched Services Management

The uSched Services are managed by the 'usched' command that is installed by default in /usr/sbin/usched. Although it is possible to manage the services directly through this command, it is advisable to install the initialization scripts and manage the services with the appropriate operating system managing interface. All the operations implemented in /usr/sbin/usched are available for the installed initialization scripts.

9.1 Starting uSched Services

The following command will start all the required uSched Services:

```
# usched start
```

9.2 Stopping uSched Services

The following command will stop all uSched services:

```
# usched stop
```

9.3 Reloading uSched Services

To reload the uSched services, execute:

```
# usched reload
```

9.4 Flushing uSched Services state

To force the serialization of the uSched Services state, execute:

```
# usched flush
```

9.5 Forcing the stop of uSched Services

If for some reason the uSched Services weren't stopped gracefully (caused by a power cut, operating system crash, etc) and the 'usched start' command isn't being able to start the services, it is possible to force stop of the uSched Services, reseting the status of the service to unitialized:

```
# usched force_stop
# usched start
```

10 uSched Services Configuration

The following subsections will describe a few common configuration examples for the uSched Services. Typically, all the other properties fit most of the systems with their default values. For more advanced configurations, refer to section **8. uSched Admin**.

10.1 Setting up remote access

The following example will enable remote access on port 7600 on all network interfaces. It will also authorize remote users to authenticate and will add a new remote user called 'myusername' with UID 1000 and GID 1000:

```
# usa change network bind addr 0.0.0.0
# usa change network bind port 7600
# usa change auth remote users 1
# usa add users myusername 1000 1000
# usa commit all
```

10.2 Changing unprivileged user and group

The following example will change the default unprivileged user and group (nobody, nogroup) to user 'someuser' and group 'somegroup':

```
# usa change core privdrop user someuser
# usa change core privdrop group somegroup
# usa commit core
```

10.3 Managing blacklists and whitelists

The following example will blacklist the UID 1000 and UID 2000, preventing any request from these UIDs to be accepted:

```
# usa change auth blacklist uid 1000,2000
# usa commit auth
```

If we want to authorize only the UID 1000 and UID 2000, rejecting request from any possible other UID, we should insert them in the whitelist:

```
# usa change auth blacklist uid ”
# usa change auth whitelist uid 1000,2000
# usa commit auth
```

If the same value is present in both white and blacklist, the blacklist takes precedence over the whitelist.