



Process Monitor Collector

Administrator Manual

Software Version 1.0.0



EducateIT GmbH

Steinenbergstrasse 7, 4532 Feldbrunnen, Switzerland

Phone: +41 32 355 39 00

E-Mail: info@educateit.ch

Web: <http://www.educateit.ch/>

Process Monitor Collector – Administrator Manual

Copyright: ©2017 by EducateIT GmbH

EducateIT GmbH
Steinenbergstrasse 7
CH-4532 Feldbrunnen
Switzerland

Phone: +41 32 355 39 00
E-Mail: info@educateit.ch
Web: <http://www.educateit.ch/>

Illustrations and Text: TE
Layout and Design: TE
Last Modification: 2017-05-04 15:08

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by and means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the EducateIT GmbH. The EducateIT GmbH assumes no responsibility for errors or omissions, or for damages that may result from the use of information (especially including example configurations and listings) contained herein.

Content

Preface	5
Features	5
See Also	5
Configuration	7
The profiles List	7
hostFile.....	11
minimumPause	11
retrieveInterval.....	11
psServicePort	11
historyDatabase.....	11
filter	13
applicationMap.....	13
network	15
secrets	16
removeProcessNameExtension.....	18
agentApiKey	18
agentApiSecret.....	18
Diagnose Problems	19
Problems Reported in Event Log	19
Enable Logging of Trace Messages	19
Starting the Service Manually	19
Using Different Configuration Files	20
Enabling Trace Sections	20
About the Host File	23
Basic Format	23
Comments and Empty Lines	23

Host Entries	23
Automatic Reload	24

Preface

“*EducateIT Process Monitor Collector*” is a service to monitor processes of any number of servers in real-time. The service keeps a map with all running processes on all servers and also stores a history, where each process and its running duration is recorded for further analysis.

Features

- Can use *PowerShell remoting* to access the process lists on any remote server. No agent installation on the servers required.
- Comes with an *Agent service* to retrieve process lists with the best performance and lowest resource usage.
- Flexible server list, which can be updated while the service is running. No service restart required.
- Builds a database with history information about all running processes.
- Allows multiple different retrieving methods to get the processes from the servers. Each server can be accessed using a different method.
- Secure *SSL* encrypted and authenticated connection between any client and the service.
- Provides a convenient *HTTPS/JSON* based interface to query a process list from scripts or other applications.
- Filters can limit the processes globally or for a single client connection to restrict the access.
- Uses the “*EducateIT Log System*”, “*EducateIT Service Framework*” and “*EducateIT Configuration B*” for best performance and stability.

See Also

The service is using the *Configuration B* system for the configuration. To understand all aspects of the configuration, it is recommended to read the document “*Configuration B - Administrator Manual*”.

The service also uses the *EducateIT Service Framework* and the *EducateIT Log System*. Both systems provide several features

which are explained in their own manuals: “*Service Framework – Administrator Manual*” and “*Log System – Administrator Manual*”

You can configure filters and an application map using our *EducateIT Filter Language* which is described in the document “*Filter Language – Administrator Manual*”.

You should get all these documents along with this manual. If not, please request them from us.

Configuration

The configuration of this service is based on “*Configuration B*” and is supporting all features of this system. The configuration is stored in the file “`configuration.xml`” which is placed in the program directory beside the executable, but can also be placed in any of the searched directories to be more OS compliant. Read the “*Configuration B*” manual, section “Files and Locations” for details.

Illustration 1 on page 8 shows the structure of the application configuration. The following sections describe each value of the configuration in detail.

There is a configuration module for the log system. You will find all details about the log system and the configuration in the “*EducateIT Log System*” manual.

The profiles List

This list defines one or more profiles to retrieve process data from a server. Each entry has a unique identifier which can be used in the hosts file to select the profile.

profiles » identifier

This is a unique identifier to select the profile from the host list. The length has to be between one and eight characters, and only lowercase letters are allowed.

profiles » retrieveMethod

Select the retrieve method which is used to get the process list from the host. Valid values are shown in the following table.

Method	Description
Agent	A dedicated agent service is used to retrieve the process list. This is the recommended method to retrieve the process list.
PowerShellNative	The native <code>Get-Process</code> command is used to retrieve the process list.
PowerShellWMI	A <i>WMI</i> command is used over <i>PowerShell</i> to retrieve the process list.

Module	Application
Value	agentApiKey Optional String
Value	agentApiPort Optional Integer
Constraints	- Minimum value: 1 - Maximum value: 65535
Default	17697
Value	agentApiSecret Optional Password
Value	applicationMap Optional String
Constraints	Has to be an absolute path to an existing file.
Value	filter Optional String
Value	historyDatabase Optional String
Value	hostFile String
Constraints	Has to be an absolute path to an existing file.
Value	minimumPause Optional Integer
Constraints	- Minimum value: 1 - Maximum value: 3600
Default	10
List	network Optional
Definition	List Entry Definition
Value	port Optional Integer - Minimum value: 0 - Maximum value: 65535 0
Constraints	
Default	
Value	protocol String - Must not be empty. - Has to be one of the following values: Message, HTTP
Constraints	
Value	scope String Has to be one of the following values: None, LocalHost, Any LocalHost
Constraints	
Default	

Illustration 1 The Application module for the service.

List	profiles
Constraints	Must not be empty.
Definition	List Entry Definition
Value	authenticationPassword Optional Password
Value	authenticationUsername Optional String
Value	identifier String
Constraints	- Minimum length: 1 character(s). - Maximum length: 8 character(s). - Has to match the following case insensitive regular expression: <code>[a-z]{1,8}</code>
Value	retrieveMethod String
Constraints	Has to be one of the following values: PowerShellNative, PowerShellWMI, Agent
Value	psServicePort Optional Integer
Constraints	- Minimum value: 1 - Maximum value: 65535
Default	17691
Value	removeProcessNameExtension Optional Boolean
Default	Yes
Value	retrieveInterval Optional Integer
Constraints	- Minimum value: 10 - Maximum value: 3600
Default	30
List	secrets
Definition	List Entry Definition
Value	authorizedNetworks Optional String
Value	authorizedProtocols Optional String
Constraints	Has to be one of the following values: Any, Message, HTTP

Default	Any
Value	authorizedServices String List
Constraints	Each element of in this list has to be one of the following values: ReadProcessList
Value	key String
Constraints	Must not be empty.
Value	label String
Constraints	Must not be empty.
Value	processFilter Optional String
Value	secret Password
Constraints	Must not be empty.

The Agent method is the recommended and best way to retrieve the process list from hosts. It requires the installation of the *Agent* service on each host, but has the best performance with the lowest resource usage. Only the agent method can retrieve the relative *CPU usage* for each process.

The PowerShellNative is the method with a passable performance, but this method only works if the server supports the -IncludeUserName argument using the Get-Process command.

If the host does not support the -IncludeUserName argument, use the PowerShellWMI method, which is slower but works across any Windows version.

You can mix the retrieve methods and use different ones for different hosts.

profiles » authenticationUsername

If you specify this optional configuration value, this user name is used as authentication for the remote host.

profiles » authenticationPassword

If you specify the authentication user name, you have to provide a password for the authentication.

Never store a plaintext password in the configuration. Use the *EducateIT Password Encoder* to protect the stored password.

Encode Passwords!

Always encode your passwords using the *EducateIT Password Encoder* tool.

hostFile

This configuration value has to be an absolute path to the file with the current host list. See [“About the Host File” on page 23](#) for details about the format of the host file.

minimumPause

This value defines the minimum pause between two process list receive tasks in seconds. This value ensures a given pause to make sure to free the *CPU* for a while on each server.

The value has to be between one and 3600 seconds. The default value is 10 seconds.

retrieveInterval

This value defines the interval to retrieve the process list from a server in seconds. Faster intervals lead to a more precise recording of the process tables, but increase the load on network and *CPU* of all servers. Slower values reduce the load but lower the precision.

The default value of 30 seconds is a good start and you should only modify the value if you also monitor the load of the network and servers. The value `minimumPause` has always priority before `retrieveInterval`. If the interval is set to 30 seconds, the minimum pause to 10 and receiving the process table takes 25 seconds, the interval will be effectively 35 seconds, because of the 10 seconds pause.

The value has to be between 10 and 3600 seconds. The default value is 30 seconds.

Important Notice #1!

The history database must be on the local file system of the server. Otherwise the behaviour may be unexpected. The system may block, crash or lose data.

psServicePort

Using this optional value use can change the port of the local *PowerShell* service.

historyDatabase

To enable the process history database feature, you have to define this optional value. It has to be an absolute path to a database file. The file do not have to exist, it is automatically created.

The history database must be on the local file system of the server.

An existing database is automatically updated to the latest version if you update the process monitor service. Make sure you backup the database before testing a new version of the process monitor

Important Notice #2!

Make sure you backup the history database before you try a new version of the process monitor collector, because the database is automatically upgraded and may be incompatible after the upgrade with previous versions.

Name	Type	Description
hostName	Text	The name of the host on which the process is running.
processId	Number	The process ID.
name	Text	The name of the process. Depending on the source, this name can have a suffix like .exe or no suffix.
path	Text	The absolute path of the process.
owner	Text	The owner of the process with the domain.
totalCpuTime	Number	The current total <i>CPU</i> time in milliseconds.
userCpuTime	Number	The current user <i>CPU</i> time in milliseconds.
workingSetSize	Number	The current working set size in bytes.
virtualSize	Number	The current virtual size in bytes.
cpuUsage	Number	The current <i>CPU</i> usage as value between 0.0 and 1.0. If this value is not available, it is set to -1.0.

Table 1 A list with all values which can be used in a filter expression.

Name	Type	Description
name	Text	The name of the process. Depending on the source, this name can have a suffix like .exe or no suffix.
path	Text	The absolute path of the process.

Table 2 A list with all valid values for the filter in the object map.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Configuration ...
  <Module name="Application">
    ...
    <Value name="filter"><![CDATA[
      <Value name="owner">
        <Or>
          <Contains caseSensitivity="CaseInsensitive">NT AUTHORITY</Contains>
          <Contains caseSensitivity="CaseInsensitive">Window Manager</Contains>
          <Contains caseSensitivity="CaseInsensitive">IIS APPPOOL</Contains>
          <MatchesRegularExpression>^\\?$.</MatchesRegularExpression>
        </Or>
      </Value>
    ]]></Value>
    ...
  </Module>
</Configuration>
```

Listing 1 A filter definition directly in the configuration file as text block.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Configuration ...
  <Module name="Application">
    ...
    <Value name="filter">c:\Test\ProcessMonitorFilter.xml</Value>
    ...
  </Module>
</Configuration>
```

Listing 2 Filters defined in an independent file.

collector. An update of the database may be incompatible with a previous version of the process monitor collector.

filter

Use this optional value to define a filter to remove unwanted process entries from the system and the database. You can either write the filter directly into this value as text or specify an absolute path to a file which contains the filter.

Please read all details how to write a filter expression in the document “*Filter Language – Administrator Manual*”.

Table 1 on page 12 shows all valid values you can use in the filter expression. The filter is applied before any application mapping, therefore the application name and group and not available for the filter expression.

Embedded Filter

See Listing 1 on page 12 how you can embed a filter directly into the configuration file. Important are the enclosing `<![CDATA[` and `]]>` elements which make sure the filter expression is parsed as text block and not as XML.

Filter in a Separate File

If you have many filter definitions you should create a separate filter file and reference it using an absolute path as shown in Listing 2 on page 13. The format of the file is described in the document “*Filter Language – Administrator Manual*”.

applicationMap

This optional value can be an absolute path to the application map file. The file has the object map format described in the document “*Filter Language – Administrator Manual*”.

Table 2 on page 12 shows all valid values for the filter in the object map. There are only two valid values: name and path.

If no entry in the application map matches the process, the application name and group will be empty. You can prevent this

```

<?xml version="1.0" encoding="utf-8" ?>
<ObjectMap
  version="1"
  xmlns="http://educateit.ch/software/FilterObjectMap/1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <ObjectGroup name="Common">
    <Object name="Explorer">
      <Value name="name">
        <Or>
          <Matches caseSensitivity="CaseInsensitive">explorer.exe</Matches>
          <Matches caseSensitivity="CaseInsensitive">explorer</Matches>
        </Or>
      </Value>
    </Object>

    <Object name="Citrix Receiver">
      <Value name="path">
        <Contains caseSensitivity="CaseInsensitive">\Citrix\Receiver\</Contains>
      </Value>
    </Object>
  </ObjectGroup>

</ObjectMap>

```

Listing 3 A simple example application map, mapping two different applications.

```

<List name="network">
  <ListEntry>
    <Value name="protocol">HTTP</Value>
    <Value name="scope">Any</Value>
  </ListEntry>
  <ListEntry>
    <Value name="protocol">Message</Value>
    <Value name="scope">Any</Value>
  </ListEntry>
</List>

```

Listing 4 The network configuration, both protocols enabled and accepting connections from any host.

behaviour if you insert a last rule into the application map which always matches the process.

The following XML document example is illustrating a last, always matching rule:

```
<Object name="Unknown">
  <Value name="name">
    <MatchesRegularExpression>.*</MatchesRegularExpression>
  </Value>
</Object>
```

See [Listing 3 on page 14](#) for an example application map. It contains one single group Common and two application mappings.

network

In the network list you can configure the different supported network protocols. Each entry in the list configures one protocol.

If a protocol is omitted, it is disabled in the server. This is the same as setting the scope of the protocol to None.

network » protocol

The required protocol value in a network entry selects the protocol which is configured. The following values are supported:

- Message
- HTTP

The Message protocol is the fast one used by the *Process Monitor* client and other systems from *EducateIT*.

The HTTP protocol is the open *HTTPS/JSON* based protocol which can be used for external and custom applications.

network » port

The optional port value can be specified to change the port to listen for connections. If you omit this value, the default ports are used:

- *Message* protocol: 17692
- *HTTP* protocol: 17694

network » scope

Using the required scope value, you can control the scope from where connections are accepted. The following values are accepted:

- None: The protocol is disabled.
- LocalHost: Only connections from local host are accepted.
- Any: Connections from any host are accepted.

You can further limit the accepted connections using the `authorizedNetworks` value for each secret entry. Here you can specify network ranges which are checked for each authentication with the server.

See [Listing 4 on page 14](#) for an example how the network list can look like. This example enabled both protocols with the default ports and accept connections from any host.

secrets

This list controls which systems can access the collected process list using the *API*. You generate secret pairs using the *EducateIT Secret Generator*. The client is the software which would like to access the process list and the server is the process monitor collector service.

You should generate a own secret pair for each software which would like to access the process list. You should also encode the secret on the client using the *EducateIT Password Encoder*. The secret on the client is like a password and you should protect it like one.

There is also the possibility to limit the access using the `authorized networks` value. If you know a client is always connecting from the IP range `10.0.0.0/24`, you can limit the access further by specify this network in the `authorizedNetworks` value.

You can also set an individual filter for each secret entry. This filter will limit the processes visible to the client accessing the server using this entry.

See [Listing 5 on page 17](#) how this list could look like. The shown key and secret values are fictive for demonstration and do not work.

secrets » label

This is a label which is used for the client in log files. The label helps to easily know which secret was used to access the *API*.

secrets » key

This is the key for the *API* secret pair. Copy this value from the generated secret pair using the *EducateIT Secret Generator*.

secrets » secret

This is the server secret generated with the *EducateIT Secret Generator*. It is unpractical to generate a client secret from the server secret. Nevertheless you should limit the access to the configuration as good as possible.

API?

API is a shortcut for Application Program Interface. In the context of the process monitor collector, this is the interface where other systems can access the current running processes and process this information in a own context.

One example is the process monitor client which accesses the *API* to display the current running processes in a table.

```

<List name="apiSecrets">
  <ListEntry>
    <Value name="label">Raptor Server</Value>
    <Value name="key">fMAtR;8Gq_BpPai/TP827;ripX1ymYBx3Es;KZ;B.1X+oJsdv3PViv-FupuJf}-</Value>
    <Value name="secret">cx(5:KS(9sIRU7+I:)6#5ok{Sc)Ghi1st/1/JWVY*H(Kx4g@IzqOGR]xA08{2 0e8ouJNULXH3fAfos4Zix3VGIm*QZxtGWhiVfmgTXc*iGkIpQf+dz4Bs0Z3T85tzSQw9AK8s8n9DL7nTpDZA==</Value>
    <Value name="authorizedNetworks">10.0.0.0/16 127.0.0.1/24</Value>
    <Value name="authorizedServices">ReadProcessList</Value>
    <Value name="authorizedProtocols">Message</Value>
  </ListEntry>
  <ListEntry>
    <Value name="label">Client</Value>
    <Value name="key">?1.RAf*|#:ZQj-Z6Zl-W_BjdoKh6mF{35EU=-_Q(N|qya1iRd+vu/-X97LQ6,AQ</Value>
    <Value name="secret">_k:({%pWZIVB0_/vE$WvuI*W*FKT,t:({/mQUf,8hE1efK8B/JM07_C8tNiX9V ducE2xi+XtUSz6e+7XpFv6FVnmnlqM4AQ07BT*PLiz7RhyEMTgABvX/DLogL0zKv1DQjyJkhIn7K1TXrue==</Value>
    <Value name="authorizedServices">ReadProcessList</Value>
    <Value name="authorizedProtocols">Message</Value>
  </ListEntry>
  <ListEntry>
    <Value name="label">HttpApiAccess</Value>
    <Value name="key">Ht-uu50NTP.sa}fMVB1sygNQ!azRJH1Hc7Vb-JQE1Grvsu!hT0zjZL(R!_!kmzH</Value>
    <Value name="secret">?tDC-p#*1[5D3MS:cpm/IH-rd@krVmvbXS/ysPg18BXfO+U'GHW4*nmH=Hj39p 175WvF42fsNv/KpSXpX/yQqZuWZD0BVTfGthyYNUrS6xSGpdkRzWnyTh570NEXonfH8Z1Q+LlprjenbA6sxYQ==</Value>
    <Value name="authorizedServices">ReadProcessList</Value>
    <Value name="authorizedProtocols">Http</Value>
    <Value name="processFilter"><![CDATA[
      <Value name="name">
        <StartsWith>svc</StartsWith>
      </Value>
    ]]></Value>
  </ListEntry>
</List>

```

Listing 5 An example how the secret list can look like.

secrets » authorizedNetworks

The optional `authorizedNetworks` value can contain a whitespace separated list of networks from where this secret can be used. The IP of each incoming connection is checked, if it is part of one of the specified networks.

secrets » authorizedProtocols

The optional `authorizedProtocols` value limits the protocols supported by the given secret entry. The following identifier are valid for this value:

- Any
- Message
- HTTP

If you omit this value, Any is used which accepts all protocols.

secrets » authorizedServices

The required value `authorizedServices` controls which services can be accessed by the used secret entry. Currently there is only one supported service:

- ReadProcessList

The `ReadProcessList` service allows to query the internal process list for process entries. You can limit this access using the `processFilter` value.

secrets » processFilter

The optional `processFilter` value defines a filter which is applied to each request using the given secret entry. If you omit this value, all processes can be read.

See “[filter](#)” on page 13 for details how to specify this filter. This value supports the same features as described in the given section.

removeProcessNameExtension

This optional configuration value controls if a process name extension like `.exe` or `.com` is removed from the process name. If you omit this value, it is set to `Yes` which removes the extensions.

Some sources provide a full process name with extension, like the *PowerShell* WMI source and the *Agent* source. Other do not provide an extension for the process name, like the *PowerShell* native source. If you mix process information from multiple sources you should remove the extension to merge different process names formats into one single process entry.

agentApiKey

This is the key which is used to connect to all agents. Make sure you generate a unique secret pair for each collector service which is accessing the agent.

agentApiSecret

This is the secret which is used to connect to all agents. Encode this secret using *EducateIT Password Encoder* tool.

Diagnose Problems

If the service does not work as expected, there are several ways how to gain additional information what the service does. It gives you some valuable hints about the root of the problem.

Problems Reported in Event Log

If a problem occurs which prevents the service from starting, a message is reported in the Windows “Event Log”. You find these messages in the “Application” section.

This is especially useful if you have problems with your configuration or with the log system. The configuration has to be read successfully before any log message is written.

Another simple way to find start-up problems is starting the service manually. See [“Starting the Service Manually”](#) on page 19 for details.

Enable Logging of Trace Messages

You should enable trace messages as first action. To enable the trace messages you just need to remove any filter from the Log section in the server configuration.

The following shows an example log configuration:

```
<Module name="Log">
  <List name="FileWriter">
    <ListEntry>
      <Value name="LogFilePath">C:/application.log</Value>
    </ListEntry>
  </List>
</Module>
```

Starting the Service Manually

You can start the service like any regular program. It allows you to be sure the service runs with the permissions of your user. You also see additional messages on the console about the startup and shutdown process of the service.

To start the service as regular program, open a console and start the program with the command line argument `--serviceTrace`:

```
ProcessMonitorCollector.exe --serviceTrace
```

Using Different Configuration Files

Especially for testing there are simple ways how you can specify different locations for the main configuration file. This should be combined with the service trace mode. Using this technique, you can prepare several test environments with different configurations, which you start using scripts.

You can set the location of the main configuration file with the command line argument `--configFile`:

```
ProcessMonitorCollector.exe --serviceTrace  
--configFile=C:\Test1\configuration.xml
```

To use this technique it is necessary to remove any configuration files from the application directory, because this location is always scanned for configuration files.

You can also specify multiple configuration files using this command line argument:

```
ProcessMonitorCollector.exe --serviceTrace  
--configFile=C:\Test2\configuration.xml  
--configFile=C:\Test2\process.conf_b.xml
```

To specify additional search paths, which are scanned for configuration file, you can use the command line argument `--configDirectory`:

```
AssistantServer.exe --serviceTrace  
--configDirectory=C:\Test3Config
```

This directory can contain configuration files with the `.conf_b.xml` suffix, or with the name `configuration.xml`.

Enabling Trace Sections

Each application can have a number of “trace sections” which are disabled by default. A such trace section writes rare used diagnosis information into the log file. This, sometimes extensive amount of data, is disabled by default to keep the log file small and tidy.

The process monitor collector service supports a number of trace sections, shown in [Table 3 on page 21](#).

Identifier	Description
HostChanges	This trace section enables additional information about the interpretation of the hosts file and changes of the host list.
HistoryChanges	This trace section enables messages about history information written to the database.
HistorySql	This trace section writes all SQL statements into the log file which are used to update the history database.
CommunicationDetails	This will enable details about the message and HTTP communication. Log entries with all requests are written to the log with details about the used parameters and filters.
RetrieveDetails	This will show details about the retrieval of the process lists from the configured hosts. It will show each attempt to retrieve the process lists and also timing information, about the duration for each retrieval. This details are independent from the method (PowerShell, Agent).
ProcessStoreDetails	This will enable details how the entries in the process store are updated. It will display log entries where you can see how many processes were added, removed or updated in the store from each host.

Table 3 A list of trace sections to get more details in the log file.

To enable a trace section you have to add them to the Log module configuration as shown in the following example. The list with the trace section identifiers is marked in yellow.

```
<Module name="Log">
  <List name="FileWriter">
    <ListEntry>
      <Value name="LogFilePath">C:/application.log</Value>
    </ListEntry>
  </List>
  <List name="EnabledTraceSections">
    <Value>HostChanges</Value>
  </List>
</Module>
```

Trace Sections Performance Impact

Make sure you never enable any trace sections for a system in a production environment. Enabling trace sections can significantly slow down a system, even if the trace messages get filtered from the log file.

About the Host File

This section is about the host file which is used to specify a list of hosts which are monitored.

Basic Format

The host file has to be a simple text file, encoded in *UTF-8*. The line ends in the file can be *LF* or *CRLF*. The basic format is line oriented: Each line contains a single host entry.

Have a look at the example file [Listing 6 on page 23](#):

Comments and Empty Lines

Empty lines, or lines starting with a '#' character are completely ignored. Use can use such lines to add comments to the file, e.g. the timestamp of the generator or similar.

Any whitespace (spaces, tabs) is removed from a read line before processing. Therefore a line is empty even it contains just a number of spaces or tabs and the '#' of a comment can be prefixed with tabs or spaces.

Host Entries

Any other line is interpreted as host entry. In the simplest form a host entry is just a host name or *IP* address. Again, any whitespace at the start and end of the line is removed before processing, therefore you can even indent the lines with the host names if required. It does not make any change.

```
#  
# This is an example file  
#  
  
example1  
example2;b  
example3;b  
example4;a
```

Listing 6 Example host file.

By adding a “;” character after the host name, followed by a profile identifier, you can select a given profile for the host entry.

Have a look at our example file [Listing 6 on page 23](#). The entry for host `example1` has no profile specified. In this case the system automatically selects the first profile from the configuration.

Both hosts `example2` and `example3` select a profile with the identifier “b” from the configuration and `example4` selects a profile with the identifier “a”.

There must be no spaces before and after the “;” character in the host file. While whitespace at the start and end of the lines are ignored, this is not the case for the profile selection.

No Spaces Around “;”!

There must be no spaces before and after the “;” character in the host file. While whitespace at the start and end of the lines are ignored, this is not the case for the profile selection.

Automatic Reload

The host file is automatically reloaded if the file modification time changes. Make sure you do not overwrite the file directly, instead use an atomic write operation. An atomic write operation is necessary to prevent the system from reading an incomplete file.

An atomic write process for the example file `hosts.txt` could look like this:

- Create a new temporary file `hosts.txt.new`.
- Write the new host list into the temporary file.
- Rename the existing file `hosts.txt` into `hosts.txt.old`.
- Rename the new file `hosts.txt.new` into `hosts.txt`.
- Delete the old file `hosts.txt.old`.

Use Atomic Write!

It is really important that you use an atomic write operation to update the host file. Otherwise the behaviour of the system may be unexpected, because the host file may be read in an incomplete state.



EducateIT GmbH
Steinenbergstrasse 7, 4532 Feldbrunnen, Switzerland
Phone: +41 32 355 39 00
E-Mail: info@educateit.ch
Web: <http://www.educateit.ch/>