GROUP.Sandbox

Parameter Description and Configuration in iQ.Suite for IBM Domino

Document version 9.0
iQ.Suite for IBM Domino as of Version 17.1

Content

1 GROUP.Sandbox in iQ.Suite for Lotus Domino

GROUP.Sandbox is an iQ.Suite solution designed to fully integrate auxiliary components such as converters, unpackers, virus scanners or spam analyzers into iQ.Suite. For each component to be integrated, a separate *GROUP.Sandbox* serves as interface to the MailGrabber or DatabaseGrabber.

Using *GROUP.Sandbox* allows to increase the stability of the system environment by separating critical systems such as virus scans or file analysis processes from the Grabber processes. Thus, a virus scanner crash or deadlock will not fatally affect the stability of the Domino server.

iQ.Suite provides an integrated sandbox solution for the following components:

Component	iQ.Suite Module
Virus scanner - integration and pattern update	iQ.Suite Watchdog
Antispam analyzer - integration and pattern update	iQ.Suite Wall
Converter	iQ.Suite Wall
Unpacker	iQ.Suite Wall / iQ.Suite Watchdog
Analyzer	iQ.Suite Wall
Integration with iQ.Suite Store	iQ.Suite Bridge

The *GROUP.Sandbox* implementation and functionalities differ according to the type of component. This document provides a description of the sandbox principle for virus scanners and virus pattern updates.

1.1 Default Ports

For the sandboxes the following port numbers are set as of iQ.Suite Version 14.1:

Sandbox	Port number	Component
soap.(n)tk_core.dll.defaults.ini	8230	CORE component
soap.(n)tk_d2xl.dll.defaults.ini	8240	File-To-XML analyzer
soap.(n)tk_norman.dll.defaults.ini	8300	Norman virus scanner
soap.(n)tk_oittext.dll.defaults.ini	8320	File-To-Text analyzer
soap.(n)tk_sasi.dll.defaults.ini **	8280	Sophos antispam engine
soap.ntk_mcafee2.dll.defaults.ini	8290	External McAfee virus scanner (as of Version 8.5)
soap.ntk_sophos2.dll.defaults.ini	8220	External Sophos virus scanner
soap.tk_ccanalyser.dll.defaults.ini	8380	Credit card analyzer
soap.tk_kaiarchive.dll.defaults.ini ***	8270	iQ.Suite Store connection via iQ.Suite Bridge
soap.tk_mcafee3.dll.defaults.ini	8390	Internal McAfee virus scanner (as of iQ.Suite 14.1)
soap.tk_savapi.dll.defaults.ini *	8340	Internal Avira virus scanner
soap.tk_sophos3.dll.defaults.ini	8400	Internal Sophos virus scanner (as of iQ.Suite 15.0)
soap.tk_unpack2.dll.defaults.ini *	8330	Unpacker

* :	Not available on AIX.
** :	Not available on AIX or Solaris.
***:	Not available on AIX, Solaris or Linux.

1.2 Return Codes

The following return codes are returned in case of errors in the sandbox-clients (*soap.tk_<virus scanner>.dll*) of a virus scanner sandbox:

Return Code	Description
200	Function call without previous initialization
201	Error on basis initialization
202	Error on initialization
203	Sandbox error (no contact/response, repeated error) and timeout
204	Exception or other error in the sandbox client
205	On-access scanner runs in a temporary directory

2 Manual Configurations

2.1 Enabling the Sandbox

By default, the sandboxes are enabled by the iQ.Suite installation process, and the sandbox files are located in the iQ.Suite data directory, e.g. `%ExecDir%\<virus scanner>`.

The virus scanners are addressed by way of the GROUP Interface DLL. Typically, no modifications are required, except for the following components:

- Virus scanners or spam analyzers that require periodical pattern updates or engine updates. Refer to Automatic Virus Pattern Update.
- Partitioned environments. Refer to Particularities for Partitioned Servers Under Unix.

2.2 Disabling the Sandbox

Normally, the configuration documents of sandbox components such as virus scanners should not be disabled, as this means the component (e.g. a virus scanner) itself is switched off as well. To be able to disable a sandbox for a short period of time (e.g. for test purposes) without disabling the virus scanner, proceed as follows:

1. Open iQ.Suite.
2. Copy the configuration document of the sandbox component to be disabled. For instance, to disable the Sophos virus scanner sandbox, select *WATCHDOG -> UTILITIES -> VIRUS SCANNER* and copy the configuration document for the Sohpos Scan Engine.
3. In the copy, under *SETTINGS -> SCAN CALL*, change the sandbox client DLL path, e.g. from `%ExecDir%\sophos3\soap.tk_sophos3.dll`
   to `%ExecDir%\sophos3\tk_sophos3.dll`.
4. Enable the copy and disable the original configuration document.

The sandbox can now be enabled or disabled by switching the configuration documents.

3 Sandbox Principle for Virus Scanners

The following sections describe the working principle and the processing sequence of each sandbox element using the example of the Sophos virus scanner (<sophos3>).

3.1 Sandbox Elements

A virus scanner sandbox includes at least the files described in the table below:

	Files - Windows	Files - Unix	Purpose
1	tk_<sophos3>.dll		GROUP Interface DLL; interface to third-party product
2	soap.tk_<sophos3>.dll	soap.tk_<sophos3>.dll	Sandbox client DLL
3	tk_<sophos3>.dll.exe	soap.tk_<sophos3>.dll.srv	Sandbox server EXE
4	soap.tk_<sophos3>.dll. defaults.ini	soap.tk_<sophos3>.dll. defaults.ini	SOAP.Defaults.INI; GBS configuration file with the sandbox default settings
5	soap.tk_<sophos3>.dll. ini	soap.tk_<sophos3>.dll.ini	SOAP.INI (optional); customizable version of the SOAP.Defaults.INI file that allows to adjust the sandbox default settings
6	tk_<sophos3>_ref.cfg	tk_<sophos3>_ref.cfg	Component-specific configuration file called by the sandbox server EXE through an update program (optional)

3.2 Processing Sequence

The sandbox server EXE runs the engine updates automatically:

1. When the Watchdog job required for the virus scanner is started, the sandbox client DLL (2) is addressed by the Grabber and automatically loaded.
2. The sandbox client DLL ensures the sandbox server EXE (3) is started. The communication between both files is performed via TCP/IP using the gSOAP implementation of the SOAP protocol[1]. During the process, the sandbox server EXE acts as sandbox server and the sandbox client DLL as sandbox client.
3. When the sandbox server EXE is started, the GROUP Interface DLL (1) is loaded, e.g. GAPI (GBS Application Programming Interface), GAVI (GBS AntiVirus Interface), unpacker etc. The interface ensures the communication with the third-party product.
4. The sandbox server EXE (3) and the sandbox client DLL (2) load the SOAP.Defaults.INI (4) and SOAP.INI (5) files and apply the sandbox settings saved in these files. (4) contains preset configurations, which are automatically overwritten by new default values every time iQ.Suite is

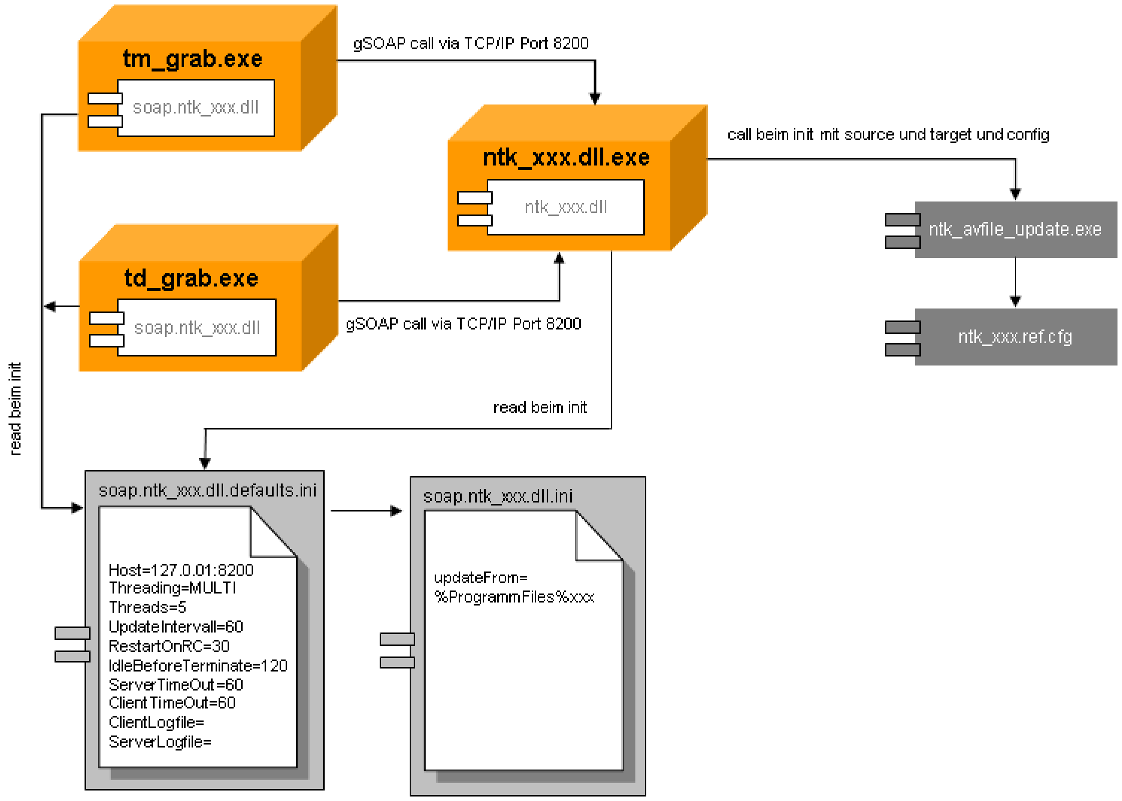
[1] For details on gSOAP refer to http://gsoap2.sourceforge.net.

updated, whereas (5) can be customized to use specific settings for your environment. The settings in (5) have priority over those in (4).

5. If a configuration file (6) is included (here: *tk_sophos3_ref.cfg*), it is also called by the sandbox server EXE through an update program.

Regardless of the number of iQ.Suite jobs configured, only one sandbox server EXE is started per sandbox. In the context of an iQ.Suite configuration including, for instance, both a mail job and a database job for a virus scanner, the sandbox server EXE is started only once and jointly used by the MailGrabber and DatabaseGrabber processes. On the other hand, the sandbox client DLL is loaded separately by both the MailGrabber and the DatabaseGrabber.

Note: If *n* different virus scanners are used on the same system, *n* sandbox server EXE instances will be started. To this end, the sandbox default configurations include *n* preconfigured TCP ports, e.g. Port 8210 for the McAfee virus scanner.



```
Host=127.0.01:8200
Threading=MULTI
Threads=5
UpdateIntervall=60
RestartOnRC=30
IdleBeforeTerminate=120
ServerTimeOut=60
ClientTimeOut=60
ClientLogfile=
ServerLogfile=
```

```
updateFrom=
%ProgrammFiles%xxx
```

3.3 SOAP.Defaults.INI and SOAP.INI Parameters

The SOAP.Defaults.INI (*soap.tk_<xxx>.dll.defaults.ini*) and SOAP.INI (*soap.tk_<xxx>.dll.ini*) files are used to configure the sandbox behavior.

The SOAP.Defaults.INI file contains default sandbox settings, including the default port number. To change the sandbox behavior, enter appropriate settings in the SOAP.INI file rather than the SOAP.Defaults.INI file. Otherwise your changes will be overwritten with the SOAP.Defaults.INI default values by the next iQ.Suite update.

The settings in the SOAP.INI file have priority over those in the SOAP.Defaults.INI file. The parameters described below can be set in both of the files.

Note: For file or directory name parameters, you can specify an absolute path or a relative one. Relative names are interpreted as relative to the parameter file.

Example:
Relative path: `%ExecDir%\sophos3\soap.tk_sophos3.dll.ini`
Log file specified: `..\logs\sophos3_<client/server>.log`
Result: The log file is saved under `%ExecDir%\logs\sophos3_<client/server>.log`.

3.3.1 General Parameters

Depending on whether the sandbox is used to address an unpacker, a virus scanner or an analyzer, different parameters will be available in this file. Typically, the following parameters are always available in each SOAP.Defaults.INI or SOAP.INI file:

- `ClientLogFile=`<Name of the log file>

  Used to log messages from the sandbox client DLL.

- `EnableCoreDump=<YES/NO>`

  (Only on Unix systems) This parameter allows trouble-shooting on sandbox server errors. Use this parameter not before consulting the GBS Support Team. By default this parameter is set to `NO`. With `YES` the file size for core is set to the maximum value and a so-called dump file is stored in the sandbox directory (settings under `ServerDirectory`). In the case of errors send this file to the GBS Support Team.

- `EnvPath=`<Name>=<Path>

  Sets the environment variable <Name> in the sandbox server EXE to the value <Path>. If no absolute path is specified, it is assumed to be relative to the SOAP.Defaults.INI or SOAP.INI file and transformed into an absolute path.

Example:
With `EnvPath=MyDir=.` the path `C:\lotus\domino\data\iQ.Suite\subdir` corresponds to `EnvVar=MyDir=C:\lotus\domino\data\iQ.Suite\subdir.`

iQ.Suite uses several Oracle converters for conversion, e.g. File-To-Text converter, File-To-XML converter, PDF converter. Per default all Oracle converters create a `.oit` directory from which the sandbox is started. To change the default file path of this `.oit` directory use the environment variable `APPDATA`.

Example:
`EnvPath=APPDATA=.`
The `.oit` directory is created in the directory the SOAP.INI is stored.

- `EnvVar=`<Name>=<Value>

  Sets the environment variable <Name> in the sandbox server EXE to the value <Value>.

- `EventLogLevel=`<Loglevel>

  Sets the log level for the event log of the sandbox.

  Possible log levels: `never`, `low`, `med`, `high`, `always`. Default: `med`.

  This parameter takes priority over the value in the Windows Registry (value of the global parameter `ToolKit_GlobalEventLogLevel`).

- `Host=`<IP address>:<TCP port number>

  IP address and TCP port number used by the sandbox. The IP address is 127.0.0.1, as only local connections are to be permitted. Different TCP ports have to be set for different sandboxes to avoid interference between them. Refer to Particularities for Partitioned Servers Under Unix.

- `LibraryPath=`<Path to desired directory>

  (Unix systems only) Allows to use additional directories in the library search path. Under Linux/Solaris: `LD_LIBRARY_PATH`, under AIX: `LIBPATH`. The default directory where the SOAP.Defaults.INI or SOAP.INI file is stored (`LibraryPath=.`). Enter the path to the directory as required.

Example:
`LibraryPath=/opt/<Virus scanner>/lib`
Multiple directories are separated by a colon.

- `ServerDirectory`=<Name of the sandbox server EXE directory>

  (optional parameter) As certain files need to be stored in the same directory, the server and client files are stored in the same directory by default. To be able to separate the server and client components, we recommend using the `ServerDirectory` parameter. If you set this parameter in one of the INI files, the INI files only need to be located in the same directory as the sandbox client DLL.

Note: Files including their own configuration parameters, e.g. log files, are not affected by this parameter. Where required, they need to be individually stored in the server directory. All other files are expected to be available in the server directory specified. More specifically, these include the GROUP Interface DLL (to be run in the sandbox) and the sandbox server EXE.
Please note that the path settings may have to be changed in other parameters as well.

- `ServerLogFile=`<Name of the log file>

  Used to log messages from the sandbox server EXE. If no absolute filename is set, both parameters are created in the directory that also contains the other sandbox files.

  Relative filenames refer to the storage location of the INI file.

  For troubleshooting purposes, you can use the `%ID%` metasymbol in the filename parameter, which is later replaced with a timestamp when the log file is created. This ensures that any existing log files will not be overwritten when the sandbox client DLL and sandbox server EXE are started. Please note that, in this case, the log files need to be deleted manually.

Example:

```
ClientLogFile=tk_sophos3_client_%ID%.log

ServerLogFile=tk_sophos3_server_%ID%.log
```

Example:
To disable logging, simply leave the filename empty:

```
ClientLogFile=

ServerLogFile=
```

- `Threading=[MULTI | SINGLE]`

  The `SINGLE` option reduces the hard disk usage.

- `Threads=`<Number of possible threads for multi-threading>

  Sets the parallel processing level in the sandbox server EXE.

3.3.2 Parameters for Temporary Directories

- `CleanTmpDir=`<Value>

  Each time the sandbox server EXE is started, the content of the directory specified under `TmpDir` is deleted. This requires that the name of directory specified under `TmpDir` is "tmp” or "temp” or the extension is ".tmp".
  Possible values: `YES, NO`. Default: `NO.`

Note: To avoid data loss configure your own exclusive directory under `TmpDir`.

- `ExtraTmpVariable=`<Name of the environment variable>

  (for exceptional cases only) This parameter allows to specify a further environment variable, to be set in addition `TEMP`, `TMP` and `TMPDIR`.

  For instance, if using the Sophos virus scanner, you can set this parameter as follows: `ExtraTmpVariable=SAV_TMP`.

  Default: `empty string`

- `OnAccessScanCheck=`<Value>

  When used with on-access virus scanners, this parameter causes the directory set under `TmpDir` to be checked. Possible values: `YES, NO`. Default: `YES.`

- `TmpDir=`<Name of the new temporary directory of the sandbox>

  This parameter complements the sandbox environment variables `TEMP`, `TMP` and `TMPDIR`. No on-access virus scanner should be scanning this directory.

  Default: `empty string`

3.3.3 Sandbox Client DLL Time Response Parameters

Timeouts of the sandbox client DLL occur when the configured period of time allocated for the processing of sandbox server EXE actions is exceeded. The sandbox server EXE actions include virus checks, spam checks, unpacking, initializing virus scanners and analyzers, etc.

Normally, the following parameters are used to configure the sandbox client DLL time response:

- `ClientTimeoutIO`=<Number of seconds until timeout by the client>

  If the sandbox server EXE exceeds the time period set here while performing an action, the sandbox client DLL stops and then restarts the sandbox server EXE.

Note: By experience, the initialization of virus scanners or analyzers can be quite time-consuming. Therefore, we recommend setting the calculated initialization as time interval. With the default setting, the operation is aborted after 90 seconds.

- `ClientTimeoutMin`=<Number of seconds until the client assumes an error>

  If any actions cannot be performed, e.g. due to a timeout caused by the `ClientTimeoutIO` parameter, due to a crash of the sandbox server EXE or due to a fatal server error, the system attempts to repeat the action.
  The two parameters `ClientTimeoutMin` and `ClientTimeoutMax` can be used to set the time interval for this attempt. With the default setting, the system attempts to perform the action for a period of 210 seconds.

Note: When this period of time expires, the sandbox client DLL signals an error, which is recorded in the Notes log. Temporary problems that do not occur again when the action is repeated, are only recorded in the sandbox client DLL and sandbox server EXE log files.

- `ClientTimeoutMax`=<Number of seconds until the client assumes an error>

  Default: 210 seconds. By default, `ClientTimeoutMin` and `ClientTimeoutMax` are set to the same value.

To have the three parameters above set automatically, use the following parameters:

- `ClientTimeout`=<Value>

  The `ClientTimeoutIO` parameter is set to the specified <Value>.

  `ClientTimeoutMin` and `ClientTimeoutMax` are set to three times the value of `ClientTimeoutIO`.

Note: With both the `ClientTimeout` parameter and one of the other parameters set, the last parameter specified in the file applies. In other words, if the last parameter in the file is `ClientTimeout`, all other parameters are ignored.

Special configuration

In certain cases, it may be useful to set `ClientTimeoutMin` to a smaller value than `ClientTimeoutMax`. In this case, the following will happen:

- When the sandbox is started, the system assumes that the sandbox server EXE is working properly (Mode 1). The timeout value used is the one specified under `ClientTimeoutMax`.
- Whenever this timeout interval is exceeded due to an action that cannot be performed, the system switches to Mode 2.
- In Mode 2, the timeout value used is the one specified under `ClientTimeoutMin`. Accordingly, if an error occurs while performing the next action, the action will be timed out sooner in case of delays.
- On the other hand, if the action is completed successfully, the system switches back to Mode 1. Successful actions include a successful virus or spam check. However, a successful initialization alone is not sufficient to switch back to Mode 1.

Sample configurations

Example 1: Timeout after 1 minute (individual action).

```
ClientTimeout=90
```

Example 2: Timeout after 1 minute (individual action) or 3 minutes (for repeated attempts).

```
ClientTimeoutIO=90
ClientTimeoutMin=210
ClientTimeoutMax=210
```

Example 3: Timeout after 20 minutes (individual action) or 3 minutes (for repeated attempts). Setting a very long timeout for individual actions and a short one for repeats can be useful when using the CORE Analyzer. As the analyzer requires more time for certain actions (such as initialization, teaching etc.), this allows to reduce the waiting time.

```
ClientTimeoutIO=1200
ClientTimeoutMin=210
ClientTimeoutMax=210
```

3.3.4 Sandbox Server EXE Time Response Parameters

The following parameters are used to configure the sandbox server EXE time response:

- `IdleBeforeTerminate`=<Number of seconds until the sandbox is terminated>

  If no actions need to be performed during the period of time set here, the sandbox server EXE is automatically terminated. Default: 120 seconds.

- `ServerTimeout`=<Number of seconds until the server times out>

  During the period of time set here, the sandbox server EXE checks whether or not it is still being used by a sandbox client DLL and the value set under `IdleBeforeTerminate` has been exceeded. Default: 60 seconds.

Note: If the following error message is recorded in the server log file, the server timeout interval has been exceeded.

SOAP FAULT: SOAP-ENV:Server
"Timeout"
Detail: TCP accept failed in soap_accept()

The configured period of time has elapsed without any action started by the sandbox server EXE. This message does not cause any malfunction of the sandbox server EXE and can be ignored.

- `TimeLimit`=<Number of minutes until the sandbox server EXE is terminated>

  After the period of time specified here has elapsed, the sandbox server EXE is terminated and restarted, regardless of the workload.

- `CallLimit`=<Max. number of sandbox requests>

  After the number of calls specified here has been reached, the sandbox server EXE is terminated and restarted, regardless of the workload. Depending on the type of the sandbox, one or more sandbox requests may be necessary for an email or document to be processed.

3.4 Automatic Virus Pattern Update

The periodical virus pattern updates required for virus checks represent a major issue, as the update procedure depends on the virus scanner used. Where required, certain virus scanner installation files are copied to the iQ.Suite directory of the corresponding virus scanner. This ensures that the original files of the virus scanner are not blocked by iQ.Suite at their original storage location. The update procedure included in the virus scanner software – or another appropriate procedure – thus ensures the files are updated as required.

Whenever new files are found while the iQ.Suite directory is periodically checked by *GROUP.Sandbox*, the update process is triggered and the virus scanner files in the sandbox are replaced. This process does not require restarting iQ.Suite Watchdog nor the MailGrabber or DatabaseGrabber. Administrators can be notified on successful and/or erroneous updates of the engine or pattern files.

The parameters required for the automatic virus pattern update are preset and usually do not have to be adjusted. Please observe the information on the particularities of each virus scanner in the sections below.

3.4.1 Particularities of Norman Virus Scanner

The virus pattern update parameters are also included in the SOAP.Defaults.INI and SOAP.INI files, in addition to the regular parameters. Refer to SOAP.Defaults.INI and SOAP.INI Parameters on Page 7.

In Windows systems, the following parameters are relevant for the update process:

- `UpdateConfig`=<Name of the configuration file>

  If you want to use another configuration file (.cfg) than the one stored in the virus scanner directory, enter this parameter manually in the SOAP.INI file and specify the name of the desired configuration file.

- `UpdateFrom`=<Source directory for the update>

  The parameter defines the path to the engine and pattern files of the virus scanner, e.g. `<ProgramDir>\Norman\Nse\Bin`.

  As an alternative you can use the Windows Registry path. In this case enter the placeholder `%RegPath%` for the `UpdateFrom` parameter and add the following parameters:

  - □ `UpdateFromRegPath` = <Registry path to the registry key of the virus scanner>
  - □ `UpdateFromRegKey` = <Registry key of the virus scanner, e.g. ‚Application Path‘>.
    The registry key contains the path to the pattern and engine files.

- `UpdateInterval`=<Update interval in minutes>

  The virus pattern update is performed when the period of time set here has elapsed.

Default: 60 minutes.

- UpdateProgram=<Program that runs the update>

  Check the path settings to the update program.

- UpdateProtocolDir=<Directory name for the update logs>

  Update logs for a successful, failed or uncompleted update of pattern or engine files are logged in the sandbox server log (ServerLogFile) by default. To save the logs to a particular directory, enter the directory path, either absolute or relative to the directory in which the sandbox configuration file is stored. Make sure that this directory is the same as defined in the global parameter ToolKit_VS_norman_UpdateProtocolDir.

The following files are additionally required for the automatic update:

- *ntk_norman_ref.cfg*

  The files listed in this configuration file are copied from the virus scanner directory. If you need to modify this file, please contact the GBS Support Team for further instructions.

- *ntk_avfile_update.bat* (Windows) or *tk_avfile_update.sh* (Unix)

  Program that calls the *ntk_avfile_update.exe* program.

- *ntk_avfile_update.exe*

  Update program located in the bin directory under the iQ.Suite program directory, which copies the files from the virus scanner directory.

> **Note:** All of the files needed for virus scanner must be located in the subdirectory of the corresponding virus scanners under the iQ.Suite program directory, e.g. <iQSuiteDir>\norman.

3.4.2 Particularities of McAfee Virus Scanner

The McAfee virus scanner is supported as external engine (mcafee2) and as integrated McAfee Scan Engine (mcafee3)[2]. The virus pattern update parameters are also included in the SOAP.Defaults.INI and SOAP.INI files, in addition to the regular parameters. Refer to SOAP.Defaults.INI and SOAP.INI Parameters on Page 7.

In Windows systems, the following parameters are relevant for the update process:

- UpdateConfig=<Name of the configuration file>

  If you want to use another configuration file (.cfg) than the one stored in the virus scanner directory, enter this parameter manually in the SOAP.INI file and specify the name of the desired configuration file.

[2] For further information on the McAfee Scan Engine please refer to the seperate document on the McAfee virus scanner. Download under http://www.gbs.com.

- UpdateFrom=<Source directory for the update>

  The parameter defines the path to the engine and pattern files of the virus scanner.

  - □ When using the external engine: <iQSuiteDir>\mcafee2.
  - □ When using the integrated Scan Engine: update\extract.

  As an alternative you can use the Windows Registry path. In this case enter the placeholder %RegPath% for the UpdateFrom parameter and add the following parameters:

  - □ UpdateFromRegPath = <Registry path to the registry key of the virus scanner>
  - □ UpdateFromRegKey = <Registry key of the virus scanner, e.g. ‚Application Path‘>.
    The registry key contains the path to the pattern and engine files.

- UpdateInterval=<Update interval in minutes>

  The virus pattern update is performed when the period of time set here has elapsed.
  Default: 60 minutes.

- UpdateProgram=<Program that runs the update>

  Check the path settings to the update program.

- UpdateProtocolDir=<Directory name for the update logs>

  Update logs for a successful, erroneous or uncompleted update of pattern or engine files are logged in the sandbox server log (ServerLogFile) by default. To store the logs in a certain directory, enter the directory absolute or relative to the directory in which the sandbox configuration file is stored. Please make sure that this directory is the same as defined in the parameter ToolKit_VS_mcafee_UpdateProtocolDir.

The following files are additionally required for the automatic update:

- *ntk_mcafee_ref.cfg*

  The files listed in this configuration file are copied from the virus scanner directory. If you need to modify this file, please contact the GBS Support Team for further instructions.

- *ntk_avfile_update.bat* (Windows) or *tk_avfile_update.sh* (Unix)

  Program that calls the *ntk_avfile_update.exe* program.

- *ntk_avfile_update.exe*

  Update program located in the bin directory under the iQ.Suite program directory, which copies the files from the virus scanner directory.

Note: All of the files needed for virus scanner must be located under <iQSuiteDir>\mcafee2 or <iQSuiteDir>\mcafee3.

3.4.3 Particularities of Avira Virus Scanner (SAVAPI3)

In Windows systems, the following parameters for the virus pattern update are set in the SOAP.Defaults.INI and SOAP.INI files:

- `DownloadFrom`=<Target address of the Avira Internet Update Manager>

  If you want to control the updates from a central server, you can use the *Avira Internet Update Manager*. A central server downloads the updates from the Internet and makes them available to each of the client computers as web server. The client computers download the updates from the central server[3].

- `UpdateConfig`=<Name of the configuration file>

  If you want to use another configuration file (.cfg) than the one stored in the virus scanner directory, enter this parameter manually in the SOAP.INI file and specify the name of the desired configuration file.

- `UpdateFrom`=<Source directory for the update>

  The parameter defines the path to the engine and pattern files of the virus scanner. The path may vary according to the installation directory of the virus scanner.

- `UpdateInterval`=<Update interval in minutes>

  The virus pattern update is performed when the period of time set here has elapsed.
  Default: 60 minutes.

- `UpdateProgram`=<Program that runs the update>

  Check the path settings to the update program.

- `UpdateProtocolDir`=<Directory name for the update logs>

  Update logs for a successful, erroneous or uncompleted update of pattern or engine files are logged in the sandbox server log (`ServerLogFile`) by default. To store the logs in a certain directory, enter the directory absolute or relative to the directory in which the sandbox configuration file is stored. Please make sure that this directory is the same as defined in the parameter `ToolKit_VS_savapi_UpdateProtocolDir`.

[3] For further information on the installation and setup of *Avira Internet Update Manager* please visit the Avira website under: www.avira.com.

3.4.4 Particularities of Sophos Virus Scanner under Windows

The Sophos virus scanner is supported as external engine (sophos2) and as integrated Sophos Scan Engine (sophos3) [4]. The parameters for the virus pattern update are also set in the SOAP.Defaults.INI and SOAP.INI files, in addition to the regular parameters. Refer to SOAP.Defaults.INI and SOAP.INI Parameters on Page 7.

Under Windows, to modify the *GROUP.Sandbox* update process, copy the following parameters from the SOAP.Defaults.INI file to the SOAP.INI file:

- `DependsOnService`=<Service name>

  Only for sophos2: Name: `SAVSERVICE`; used under Win32 platforms only to be able to check the status of the update service.

- `RestartonRc`=<Return value of the virus scanner>

  Only for sophos2: This return value is used to restart the sandbox (value: 547).

- `UpdateFrom`=<Source directory for the update>

  The parameter defines the path to the engine and pattern files of the virus scanner. The path may vary according to the installation directory of the virus scanner.

- `UpdateInterval`=<Update interval in minutes>

  The virus pattern update is performed when the period of time set here has elapsed.
  Default: 60 minutes.

Note: All of the files needed for the Sophos virus scanner must be stored under `<iQSuiteDir>\sophos2` or `<iQSuiteDir>\sophos3`.

Running the update (sophos2)

As no virus check must be run during the update process, the **SavService.exe** Sophos service is stopped and not restarted before the update process is complete. Under Win32 platforms, the status of the service is periodically checked and the Sophos Scan Engine is only initialized (started) when the service is available.

In addition, the software checks a specific return code. When an update has been performed, the Sophos Scan Engine returns the value 547 to indicate that the sandbox server EXE using the virus scanner needs to be restarted. This is controlled through the `RestartOnRc` parameter in the SOAP.Defaults.INI file. Whenever this value is returned by the virus check, the sandbox server EXE is terminated and automatically restarted when the next virus check is performed.

Note: The update procedure for the integrated Sophos Scan Engine (sophos3) is described in a separate document. Download under http://www.gbs.com.

[4] For further information on the Sophos Scan Engine please refer to the seperate document for the Sophos virus scanner. Download under http://www.gbs.com.

3.4.5 Particularities of Sophos Virus Scanner under Unix

The parameters for the virus pattern update are set in the SOAP.Defaults.INI and SOAP.INI files. To modify the update procedure, refer to the instructions provided under Particularities of McAfee and Norman.

The following parameters are used to control the update process:

- `UpdateFrom`=<Source directory for the virus scanner update>

  This parameter must be set otherwise no update will be possible. The path may vary according to the installation directory of the virus scanner.
- The Sophos library directory must not be included in the `LD_LIBRARY_PATH` (Linux/Solaris) or `LIBPATH` (AIX) environment variables.

Note: All of the files needed for the Sophos virus scanner must be stored under `<iQSuiteDir>\sophos2` or `<iQSuiteDir>\sophos3`.

Running the update

Under Unix, the virus pattern update is performed through shell scripts rather than the Sophos service. The virus patterns are automatically updated by Sophos and stored in the Sophos installation directory. The configuration file *ntk_sophos_ref.cfg* copies the *libsavi.so.3* (Linux/Solaris) or *libsavi.a* (AIX) file to the iQ.Suite program directory. In addition, the software checks the directory for updated virus patterns. If that is the case, the sandbox is restarted and the new virus patterns are used for virus checking.

4 Particularities for Partitioned Servers under Unix

If several server instances with different Unix User IDs are run on a Unix system, the sandbox server EXE runs with the User ID of the sandbox client DLL that started the sandbox server EXE. As the sandbox server EXE can be started by any sandbox client DLL, the User ID may vary.

On the other hand, a User ID may also be assigned to temporary files, if they contain emails or parts of an email and are checked by a DLL running in the sandbox server EXE. Together, these two factors result in the following requirements and restrictions:

- All server instances must have write access to the directories where the sandbox log files and temporary files are stored.
- All server instances must have read access to the temporary files of all other server instances.
- Depending on the implementation of the cleaning function, it may be necessary that all server instances have write access to the temporary files of all other server instances.
- In case a sandbox server EXE freezes, it can only be terminated by the sandbox client DLL it was started by.

To avoid these problems, allocate a separate sandbox to each server instance. Adjust the TCP port accordingly in each SOAP.INI file by setting the `Host` parameter to a consecutive number for each server instance. For instance, if `Host=127.0.0.1:8200` is preset in each SOAP.Defaults.INI file, enter `Host=127.0.0.1:8201` for the second server instance and `Host=127.0.0.1:8202` for the third server instance.

5 About GBS

GROUP Business Software is a leading supplier of solutions and services for the IBM and Microsoft collaboration platforms. With the Competence Centers Security, Modernization, Mobility and Portal & BPM, GBS enables its customers to manage the challenges of today and tomorrow faster, easier and more efficiently. Over 5,000 customers and more than 4 million users worldwide trust in GBS expertise. The company operates in Europe, North America and Asia. The European headquarters is located in Frankfurt/Germany, and the North American headquarters is based in Atlanta.

Our product descriptions are of a general and descriptive nature only. They do not stipulate any specific features nor do they represent any form of warranty or guarantee. We reserve the right to change the specifications and design of our products without notice at any time, in particular in order to keep abreast of technical developments.

The information contained in this document presents the topics from the viewpoint of GROUP Business Software AG at the time of publishing. Since GROUP Business Software AG needs to be able to react to changing market requirements, this is not an obligation for GROUP Business Software AG and GROUP cannot guarantee that the information presented in it is accurate after the publication date.

This document is intended for information purposes only. GROUP Business Software AG does not extend warranty for this document, in either explicit or implied form. This also applies to quality, execution, standard commercial practice or suitability for a particular purpose.

All the product and company names that appear in this document may be trademarks of their respective owners.

Web site:	www.gbs.com
Email address:	info@de.gbs.com
Locations:	www.gbs.com/en/locations