



**Assureon<sup>®</sup>**  
**API Guide**

**Publication information**

© 2019 Nexsan Technologies, Inc.  
All rights reserved.

**Published by:**

Nexsan Technologies Canada Inc.  
1405 Trans-Canada Highway, Suite 300  
Dorval, QC. H9P 2V9. CANADA  
www.nexsan.com

**Assureon API**

Publication date: April 29, 2019

**Trademarks**

Assureon is a registered trademark of Nexsan Technologies, Inc. SATABlade, SATABoy and SATABeast are trademarks of Nexsan Technologies. Microsoft, Microsoft Windows, Microsoft Internet Explorer, Microsoft SQL Server, and Microsoft Visual Studio .NET are registered trademarks of Microsoft Corporation.

**Patents**

This product is protected by one or more of the following patents, and other pending patent applications worldwide: United States patents US7,801,871, US8,086,578  
United Kingdom patents GB2296798B, GB2297636B

**About This Document**

Unauthorized use, duplication, or modification of this document in whole or in part without the written consent of Nexsan Technologies, Inc. is strictly prohibited.

Nexsan reserves the right to make changes to this manual, as well as the equipment and software described in this manual, at any time without notice. This manual may contain links to Web sites that were current at the time of publication, but have since been moved or become inactive. It may also contain links to sites owned and operated by third parties. Nexsan is not responsible for the content of any such third-party site.

## Table of Contents

|  |    |
|--|----|
| Overview .....                                     | 6  |
| Client Side API .....                              | 7  |
| Server Side API (ILM API) .....                    | 8  |
| SignatureId form .....                             | 8  |
| Method: CheckIntegrityAuditStatus .....            | 9  |
| Method: CreateClassification .....                 | 10 |
| Method: CreateRetentionRule .....                  | 11 |
| Method: DeleteClientConfiguration .....            | 12 |
| Method: DeleteRetentionRule .....                  | 13 |
| Method: DisposeFilesByFilter .....                 | 14 |
| Method: DisposeFilesByList .....                   | 15 |
| Method: DisposeNow .....                           | 16 |
| Method: ExportCRTToTargetFolder .....              | 17 |
| Method: GetClassifications .....                   | 18 |
| Method: GetCurrentEventLogs .....                  | 19 |
| Method: GetDispositionCandidates .....             | 20 |
| Method: GetDispositionCandidatesNoVersioning ..... | 21 |
| Method: GetDispositionLogDetails .....             | 22 |
| Method: GetDispositionLogSummary .....             | 23 |
| Method: GetFileListByFilter .....                  | 24 |
| Method: GetIntegrityAuditDetails .....             | 25 |
| Method: GetIntegrityAuditSummary .....             | 26 |
| Method: GetMD5AndSHA1ByList .....                  | 27 |
| Method: GetReadAccessLogsByFilter .....            | 28 |
| Method: GetReadAccessLogsByList .....              | 29 |
| Method: GetReports .....                           | 30 |
| Method: GetRestoreDetails .....                    | 31 |
| Method: GetRestoreSummariesByMonth .....           | 32 |
| Method: GetRestoreSummary .....                    | 33 |
| Method: GetRetentionRules .....                    | 34 |
| Method: GetSavedEventLogs .....                    | 35 |
| Method: ManageIntegrityAudit .....                 | 36 |
| Method: OverrideDispositionByFilter .....          | 37 |
| Method: OverrideDispositionByList .....            | 38 |
| Method: RestoreFilesByFilter .....                 | 39 |
| Method: RestoreFilesByList .....                   | 40 |
| Method: SetBlockDispositionStatusByFilter .....    | 41 |
| Method: SetBlockDispositionStatusByList .....      | 42 |
| Method: SetReadAccessByFilter .....                | 43 |

ASSUREON API

|  |    |
|--|----|
| Method: SetReadAccessByList.....               | 44 |
| Method: UploadClientConfiguration .....        | 45 |
| Appendix – Advanced - Configuration Files..... | 46 |

## Contacting Nexsan

For questions about Nexsan products, please visit the [Nexsan support](#) Web page, and the Nexsan Assureon [Documents and Downloads](#) page. If you are unable to find the answer to your question there, please see our contact information below.

### **Service and support**

Nexsan's Technical Services Group provides worldwide assistance with installation, configuration, software support, warranty, and repair for all Nexsan products. A variety of service and support programs are available to provide you with the level of coverage and availability your operation requires.

Nexsan Assureon Documents & Downloads page:  
[https://helper.nexsansupport.com/asu\\_downloads.html](https://helper.nexsansupport.com/asu_downloads.html)

Contact Nexsan Assureon support:  
[https://helper.nexsansupport.com/asu\\_support](https://helper.nexsansupport.com/asu_support)

Worldwide Web site:  
[www.nexsan.com](http://www.nexsan.com)

## Overview

The Assureon API is divided into two components: a client side API and a server side API.

The client side API provides functions to trigger the archiving of files from a client system. The client side API is implemented using .NET Remoting.

The server side API provides functions to administer the Assureon System. Administration includes tasks such as blocking read access, disposing of files or restoring files. The server side API is implemented using Web services.

## Client Side API

The following describes the Assureon client side API:

```
public bool Ping(out DateTime serverDateUTC)
```

Pings the FSW service to check if it's alive.

```
public bool SynchronizeDirectory(string syncRule, bool initialSync)
```

Synchronize a directory with the server.

**syncRule** – Name of rule to use

**initialSync** - flag that determines if the archive bit needs to be taken into consideration when synchronizing. If true, all files are processed regardless of archive bit.

```
public bool SynchronizeDirectory(string syncRule, bool initialSync,  
byte[] ruleFile)
```

Synchronize a directory with the server.

**syncRule** – Name of rule to use

**initialSync** - flag that determines if the archive bit needs to be taken into consideration when synchronizing. If true, all files are processed regardless of archive bit.

**ruleFile** – content of rule file to use. The rule file is an XML document containing a definition of the rule.

```
public bool GetCurrentStatus(out string currentRule, out string  
currentStatus, out string currentFile, out string currentDirectory,  
out long filesScanned, out long filesSentToArchive, out long  
directoriesProcessed, out long bytesScanned, out Status statusFlag)
```

Returns the status of the synchronization in progress.

```
public void StopCurrentSync()
```

Aborts the synchronization in progress.

```
public bool GetCurrentRules(out string[] ruleNames)
```

Returns a list of the rules associated with the current rule file.

## Server Side API (ILM API)

The Assureon ILMAPI web service provides most of the functionality found in the Assureon Systems Administration Web interface.

All ILM API methods are protected using the feature management framework. A user can only call an API method if he belongs to the ILM Active Directory group that enables the specific feature.

The following functionality is supported in the ILMAPI web service:

- Creating and deleting retention rules
- Uploading and deleting watches
- Retrieving and saving disposition candidates
- Overriding retention dates
- Disposing of files
- Allow or block disposition of specific files
- Retrieving a list of files on the server
- Create a classification
- Allow or block read access to specific files
- Get current and saved event logs
- Restore files to a network location

A complete reference of all methods is found below.

**Note** For all Boolean data types, the expected values are 0 or 1.

For `DateTime`, the expected format is: `YYYY-MM-DDTHH:MM:SSZ`

### ***SignatureId form***

The `signatureId` tag specified in the XML fragments discussed in this guide are in the form:

```
Fsorganization.fsguid.standardFsguid
```

For example:

```
IBM.3fb1f4b7-5862-46dc-8d33-93c7d5c7fd86.EVERYONECL.EVERYONESCA.00000011
```



**Method:** CheckIntegrityAuditStatus

**Description:** Gets the status of the specified servers.

**Syntax:** bool CheckIntegrityAuditStatus(string hostNames, ref string xmlAuditStatus, ref string xmlResult)

**Arguments:**

**hostnames** - a comma separated list of servers where the audit server is running

**xmlAuditStatus** - details about the audit, per host

```
<auditStatus>
  <status>
    <host>{String}</host>
    <running>{Boolean}</running>
    <isSuspended>{Boolean}</isSuspended>
    <isAuditing>{Boolean}</isAuditing>
  </status>
  ...
</auditStatus>
```

**xmlResult** – result of the call

```
<result>
  <errorMessage>String</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** CreateClassification  
**Description:** Creates a new classification.  
**Syntax:** bool CreateClassification(string xmlClassification, ref string xmlResult)

**Arguments:**

**xmlClassification** - details about the classification

```
<access>  
  <fsorganization>{String}</fsorganization>  
  <classification>{String}</classification>  
  
  <subClassification>{String}</subClassification>  
  <activeFrom>{DateTime}</activeFrom>  
  <activeTo>{DateTime}</activeTo>  
  
  <enableFlexibleRetention>{Boolean}</enableFlexibleRetention>  
</access>
```

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** CreateRetentionRule  
**Description:** Creates a new retention rule.  
**Syntax:** bool CreateRetentionRule(string xmlRetentionRule, ref string xmlResult)

**Arguments:**

**xmlRetentionRule** – a description of the retention rule

```
<rule ID="my rule">  
  <fsorganization>{String}</fsorganization>  
  <minRetentionPeriod>{Int}</minRetentionPeriod>  
  <maxRetentionPeriod>{Int}</maxRetentionPeriod>  
  
  <useLastAccessedDate>{Boolean}</useLastAccessedDate>  
</rule>  
  
  <compress>{Boolean}</compress>  
  <encrypt>{Boolean}</encrypt>  
  <offset>{Int}</offset>  
</rule>
```

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** DeleteClientConfiguration  
**Description:** Deletes a client configuration.  
**Syntax:** bool DeleteClientConfiguration(string computer, ref string xmlResult)

**Arguments:**

**computer** – name of the computer including the domain (domain\computer)

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** DeleteRetentionRule  
**Description:** Deletes a retention rule.  
**Syntax:** bool DeleteRetentionRule(string xmlRetentionRule, ref string xmlResult)

**Arguments:**

**xmlRetentionRule**– xml fragment containing the retention rule id and the fsorganization.

```
<rule ID="{String}">  
  <fsorganization>{String}</fsorganization>  
</rule>
```

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** DisposeFilesByFilter

**Description:** Dispose files using a filter. A new disposition job is created and ran at the next disposition schedule.

**Syntax:** `bool DisposeFilesByFilter(string xmlFilter, string dispositionName, ref string xmlResult)`

**Arguments:**

**xmlFilter** – xml fragment containing the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
  <fsorganization>{String}</fsorganization>
  <fsid>{String}</fsid>
  <from>{DateTime}</from>
  <to>{DateTime}</from>
  <classification>{String}</classification>

  <subclassification>{String}</subclassification>
  <signatureID>{String}</signatureID>
  <computer>{String}</computer>
  <directory>{String}</directory>
  <filename>{String}</filename>
</filter>
```

**dispositionName** – name of the disposition job to create

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** DisposeFilesByList  
**Description:** Dispose a list of files. A new disposition job is created and ran at the next disposition schedule.

**Syntax:** `bool DisposeFilesByList(string xmlFiles, string dispositionName, ref string xmlResult)`

**Arguments:**

**xmlFiles** – xml fragment containing a list of signature Ids

```
<filter>  
  <signatureID>{String}</signatureID>  
  ...  
</filter>
```

**dispositionName** – name of the disposition job to create.

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** DisposeNow

**Description:** Execute disposition now.

**Syntax:** bool DisposeNow(string xmlDisposeNow, ref string xmlResult)

**Arguments:**

**xmlDisposeNow –**

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <dispositionname>{String}</dispositionname>
</filter>
```

**xmlResult – result of the call**

```
<result>
    <errorMessage>{String}</errorMessage>
    <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.



**Method:** ExportCRTToTargetFolder  
**Description:** Exports the specified report.  
**Syntax:** void ExportCRTToTargetFolder(string selectedOrganization,  
string selectedFileSystem, string exportLocation, int  
reportID, string exportDocType)

**Arguments:**

**selectedOrganization** – Name of the organization.

**selectedFileSystem** – Name of the file system.

**exportLocation** – The folder that will contain the exported report.

**reportID** – The id or the number of the report which are:  
1,4,5,7,8,9.

**exportDocType** – Which are : pdf, msword, msexcel

**Method:** GetClassifications  
**Description:** Gets the list of classifications.  
**Syntax:** bool GetClassifications(string fSOrganization, bool showSystemClassifications, ref string xmlClassifications, ref string xmlResult)

**Arguments:**

**fSOrganization** – Contains the organization name.

**showSystemClassifications** – Retrieve the system classification or not.

**xmlClassifications** – the classifications.

```
<classifications>
  <classification>
    <classification>{String}</classification>

    <subclassification>{String}</subclassification>
    <startDate>{DateTime}</startDate>
    <stopDate>{DateTime}</stopDate>

    <doNotDisposeBeforeFlag>{Boolean}</doNotDisposeBeforeFlag>
  </classification>
</classifications>
```

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetCurrentEventLogs  
**Description:** Gets the current event log.  
**Syntax:** bool GetCurrentEventLogs(ref string xmlEvents, ref string xmlResult)

**Arguments:**

**xmlEvents** - the events in xml dataset format

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetDispositionCandidates  
**Description:** Gets a list of disposition candidates  
**Syntax:** bool GetDispositionCandidates(string xmlFilter, ref string xmlCandidates, ref string lastSignatureID, ref string fileType, ref string xmlResult)

**Arguments:**

**xmlFilter** – xml fragment containing the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <from>{DateTime}</from>
    <to>{DateTime}</from>
    <classification>{String}</classification>

    <subclassification>{String}</subclassification>
    <signatureID>{String}</signatureID>
    <computer>{String}</computer>
    <directory>{String}</directory>
    <filename>{String}</filename>
</filter>
```

**xmlCandidates** - the returned list of disposition candidates

```
<dispositionCandidates>
    <file>
        <name>{String}</name>
        <computerName>{String}</computerName>
        <signatureID>{String}</signatureID>
        <size>{Int}</size>
        <dateModified>{DateTime}</dateModified>
        <expiryDate>{DateTime}</expiryDate>
    </file>
    ...
</dispositionCandidates>
```

**lastSignatureID** - last signature ID that was returned from the server. This is used for batching. The function will return the results that are 'greater' than the last signatureID, and will return an empty string when the last result was returned.

**fileSystemType** –the next file system to process.  
 A value of "done" indicates that there are is no more file systems to process.

**xmlResult** – result of the call

```
<result>
    <errorMessage>{String}</errorMessage>
    <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetDispositionCandidatesNoVersioning  
**Description:** Gets a list of disposition candidates, without versioning  
**Syntax:** bool GetDispositionCandidatesNoVersioning(string xmlFilter, ref string xmlCandidates, ref string lastSignatureID, ref string fileType, ref string xmlResult)

**Arguments:**

**xmlFilter** – xml fragment containing the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <from>{DateTime}</from>
    <to>{DateTime}</from>
    <classification>{String}</classification>

    <subclassification>{String}</subclassification>
    <signatureID>{String}</signatureID>
    <computer>{String}</computer>
    <directory>{String}</directory>
    <filename>{String}</filename>
</filter>
```

**xmlCandidates** - the returned list of disposition candidates

```
<dispositionCandidates>
    <file>
        <name>{String}</name>
        <computerName>{String}</computerName>
        <signatureID>{String}</signatureID>
        <size>{Int}</size>
        <dateModified>{DateTime}</dateModified>
        <expiryDate>{DateTime}</expiryDate>
    </file>
    ...
</dispositionCandidates>
```

**lastSignatureID** - last signature ID that was returned from the server. This is used for batching. The function will return the results that are 'greater' than the last signatureID; will return an empty string when the last result was returned.

**fileSystemType** –the next file system to process.  
 A value of “done” indicates that there are is no more file systems to process.

**xmlResult** – result of the call

```
<result>
    <errorMessage>{String}</errorMessage>
    <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetDispositionLogDetails  
**Description:** Gets the details for the specified disposition log.  
**Syntax:** bool GetDispositionLogDetails(string xmlInput, ref string xmlDispdetails, ref string xmlResult)

**Arguments:**

xmlInput –

```
<filter>
  <fsorganization>{String}</fsorganization>
  <fsid>{String}</fsid>
  <dispoid>{String}</dispoid>
  <dispodate>{String}</dispodate>
</filter>
```

**xmlDispDetails** - the resulting details

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetDispositionLogSummary  
**Description:** Gets a list of dispositions for the specified date range  
**Syntax:** bool GetDispositionLogSummary(string xmlDisposition, ref string xmlDispLogs, ref string xmlResult)

**Arguments:**

**xmlDisposition** –  
<filter>  
    <fsorganization>{String}</fsorganization>  
    <from>{DateTime}</from>  
    <to>{DateTime}</from>  
</filter>  
**xmlDispLogs** - the resulting disposition logs  
**xmlResult** – result of the call  
    <result>  
        <errorMessage>{String}</errorMessage>  
        <errorCode>{Int}</errorCode>  
    </result>

An error code of 0 signifies a successful completion.

**Method:** GetFileListByFilter  
**Description:** Get a list of files by filter.  
**Syntax:** bool GetFileListByFilter(string xmlFilter, ref string xmlFileList, ref string lastSignatureID, ref string fileType, ref string xmlResult)

**Arguments:**

**xmlFilter** – xml fragment containing the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <from>{DateTime}</from>
    <to>{DateTime}</from>
    <classification>{String}</classification>
    <subclassification>{String}</subclassification>
    <signatureID>{String}</signatureID>
    <computer>{String}</computer>
    <directory>{String}</directory>
    <filename>{String}</filename>
</filter>
```

**xmlFileList** – returned list of files

```
<fileList>
    <file>
        <name>{String}</name>
        <computerName>{String}</computerName>
        <signatureID>{String}</signatureID>
        <size>{Int}</size>
        <dateModified>{DateTime}</dateModified>
        <expiryDate>{DateTime}</expiryDate>
    </file>
    ...
</fileList>
```

**lastSignatureID** - last signature ID that was returned from the server. This is used for batching. The function will return the results that are 'greater' than the last signatureID, and will return an empty string when the last result was returned.

**fileSystemType** –the next file system to process.

If the value is "done" this means there is no more file system to process, we reach the end.

**xmlResult** – result of the call

```
<result>
    <errorMessage>{String}</errorMessage>
    <errorCode>{Int}</errorCode>
</result>
```



**Method:** GetIntegrityAuditDetails  
**Description:** Gets the details of the specified audit.  
**Syntax:** `bool GetIntegrityAuditDetails(string auditId, DateTime auditDate, ref string xmlAuditDetails, ref string xmlResult)`

**Arguments:**

**auditId** - the id of the audit

**auditDate** - the date the audit was created

**xmlAuditDetails** - the audit details

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetIntegrityAuditSummary  
**Description:** Gets the audit summaries for the specified date range.  
**Syntax:** bool GetIntegrityAuditSummary(DateTime from, DateTime to, ref string xmlAuditSummary, ref string xmlResult)

**Arguments:**

**from** - the start date

**to** - the end date

**xmlAuditSummary** - the audit summary

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetMD5AndSHA1ByList  
**Description:** Gets the MD5 and the SHA1 for the list of signature ids provided.  
**Syntax:** bool GetMD5AndSHA1ByList(string xmlASigIds, ref string xmlMD5AndSHA1, ref string xmlResult)

**Arguments:**

**xmlASigIds** - a list of signature IDs

```
<filter>
  <signatureID>{String}</signatureID>
  ...
</filter>
```

**xmlMD5AndSHA1** - The result in the following format:

```
<fileHash>
  <signatureID>{String}</signatureID>
  <md5Hash>{String}</md5Hash>
  <shalHash>{String}</shalHash>
</fileHash>
```

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetReadAccessLogsByFilter

**Description:** Gets a list of access logs by filter.

**Syntax:** bool GetReadAccessLogsByFilter (string xmlFilter, ref string xmlReadAccess, ref string lastSigID, ref string fileType, ref string xmlResult)

**Arguments:**

**xmlFilter** – xml fragment containing the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
  <fsorganization>{String}</fsorganization>
  <fsid>{String}</fsid>
  <from>{DateTime}</from>
  <to>{DateTime}</from>
  <classification>{String}</classification>
  <subclassification>{String}</subclassification>
  <signatureID>{String}</signatureID>
  <computerName>{String}</computerName>
  <directory>{String}</directory>
  <filename>{String}</filename>
  <username>{String}</username>
</filter>
```

**xmlReadAccess** – returned list of files

**lastSigID** – SignatureId of the last file processed

**fileSystemType** –the next file system to process

If the value is “done” this means there is no more file system to process, we reach the end.

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

**Method:** GetReadAccessLogsByList  
**Description:** Gets the read access logs for the specified files.  
**Syntax:** `bool GetReadAccessLogsByList(string xmlFiles, ref string xmlReadAccess, ref string xmlResult)`

**Arguments:**

**xmlFiles** – the list of files to retrieve

```
<filter>  
    <signatureID>{String}</signatureID>  
    ...  
</filter>
```

**xmlReadAccess** – the resulting files

**xmlResult** – result of the call

```
<result>  
    <errorMessage>{String}</errorMessage>  
    <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetReports  
**Description:** Gets the list of reports.  
**Syntax:** bool GetReports(ref string xmlReports, ref string xmlResult)

**Arguments:**

**xmlReports** – a list of reports

```
<reports>
  <report>
    <title>{String}</title>
    <id>{Int}</id>
    <description>{String}</description>
    <dbJob>{Int}</dbJob>
  </report>
  ...
</reports>
```

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetRestoreDetails  
**Description:** Returns details about the restore request status.  
**Syntax:** bool GetRestoreDetails(string xmlRestore, string batchId, ref string xmlLog, ref string xmlResult)

**Arguments:**

**xmlRestore –**

```
<filter>  
  <fsorganization>{String}</fsorganization>  
  <fsid>{String}</fsid>  
  <restoreid>{String}</restoreid>  
  <batchid>{String}</batchid>  
</filter>
```

**batchId** – ID of batch to query. To get a list of batch IDs, see GetRestoreSummary.

**xmlLog** - the log file that was generated for this restore operation

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetRestoreSummariesByMonth  
**Description:** Returns details about the restore request status by month.  
**Syntax:** bool GetRestoreSummariesByMonth(string xmlInput, ref string xmlSummaries, ref string xmlResult)

**Arguments:**

**xmlInput –**

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <restoreid>{String}</restoreid>
    <year>{String}</year>
    <month>{String}</month>
</filter>
```

**xmlSummaries – a list of all restores in the given month**

```
<restores>
  <restore>
    <batchIds>{Int}</batchIds>
    <numRestored>{Int}</numRestored>
    <numFailed>{Int}</numFailed>
    <numToRestore>{Int}</numToRestore>
    <restoreId>{Int}</restoreId>

    <keepDirStructure>{Boolean}</keepDirStructure>

    <restoreAllVersions>{Boolean}</restoreAllVersions>

    <overwriteExisting>{Boolean}</overwriteExisting>
    <startTime>{DateTime}</startTime>
    <stopTime>{DateTime}</stopTime>
    <restorePath>{String}</restorePath>
  </restore>
  ...
</restores>
```

**xmlResult – result of the call**

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.



**Method:** GetRestoreSummary  
**Description:** Returns a summary of the restore request status  
**Syntax:** bool GetRestoreSummary(string xmlRestore, ref string xmlSummary, ref string xmlResult)

**Arguments:**

**xmlRestore –**

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <restoreid>{String}</restoreid>
</filter>
```

**xmlSummary –** a summary of the restore operation

```
<restore>
    <numRestored>{Int}</numRestored>
    <numFailed>{Int}</numFailed>
    <numToRestore>{Int}</numToRestore>
    <status>{Int} 0=NotStarted,
                1=Restoring,2=Finished,3=DoesNotExist</status>
    <batchIds>{Int} A comma separated list of batchIds that were used
    for this restore. Use this in the call to GetRestoreDetails </batchIds>
</restore>
```

**xmlResult –** result of the call

```
<result>
    <errorMessage>{String}</errorMessage>
    <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetRetentionRules  
**Description:** Gets the list of retention rules.  
**Syntax:** bool GetRetentionRules(string fsorganization, ref string xmlRetentionRules, ref string xmlResult)

**Arguments:**

**fsorganization:** - Name of an organization

**xmlRetentionRules** – the retention rules

```
<retentionRules>
  <retentionRule>
    <retentionId>{String}</retentionId >
    <numberOfDays>{Int}</numberOfDays>
    <compressData>{Boolean}</compressData>
    <encryptData>{Boolean}</encryptData>
    <dateCreated>{DateTime}</dateCreated>

    <doNotDisposeBefore>{DateTime}</doNotDisposeBefore>

    <setExpiryFromFileDate>{Boolean}</setExpiryFromFileDate>
      <maxVersionNumber>{Int}</maxVersionNumber>
      <setReadOnlyLock>{Boolean}</setReadOnlyLock>
    </retentionRule>
  </retentionRules>
```

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** GetSavedEventLogs  
**Description:** Gets saved event logs.  
**Syntax:** `bool GetSavedEventLogs(DateTime startDate, DateTime endDate, ref string xmlEvents, ref string xmlResult)`

**Arguments:**

**startDate** – date of first event to return

**endDate** – date of last event to return

**xmlEvents** - the returned events

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** ManageIntegrityAudit  
**Description:** Starts, stops, pauses or resumes the audit process on the specified machine.  
**Syntax:** bool ManageIntegrityAudit(string xmlRequests, ref string xmlAuditResults, ref string xmlResult)

**Arguments:****xmlRequests** - the action to take, per server

```
<auditAction>
  <audit hosts= "{String}"
    action="{Int} (0=start,1=stop,2=resume,3=stop)"/>
  ...
</auditAction>
```

**xmlAuditResults** - the results of the request, per server

```
<auditRequests>
  <request host= "{String}"
    action="{Int} (0=start,1=stop,2=resume,3=stop)"
    success="{String}true/false"/>
  ...
</auditRequests>
```

**xmlResult** - result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** OverrideDispositionByFilter  
**Description:** Override the retention period of assets by filter.  
**Syntax:** bool OverrideDispositionByFilter (DateTime newExpiryDate, string xmlFilter, ref string xmlResult)

**Arguments:**

**newExpiryDate** – new expiry date to set

**xmlFilter** – the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
  <fsorganization>{String}</fsorganization>
  <fsid>{String}</fsid>
  <from>{DateTime}</from>
  <to>{DateTime}</from>
  <classification>{String}</classification>

  <subclassification>{String}</subclassification>
  <signatureID>{String}</signatureID>
  <computer>{String}</computer>
  <directory>{String}</directory>
  <filename>{String}</filename>
</filter>
```

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** OverrideDispositionByList  
**Description:** Override the retention period of a list of assets.  
**Syntax:** bool OverrideDispositionByList (DateTime newExpiryDate, string xmlFiles, ref string xmlResult)

**Arguments:**

**newExpiryDate** – new expiry date to set

**xmlFiles** – a list of signature IDs

```
<filter>  
  <signatureID>{String}</signatureID>  
  ...  
</filter>
```

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** RestoreFilesByFilter  
**Description:** Restores a set of files by filter.  
**Syntax:** bool RestoreFilesByFilter(string xmlFilter, string xmlRestoreOptions, ref string restoreID, ref string xmlResult)

**Arguments:**

**xmlFilter** – xml fragment containing the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <from>{DateTime}</from>
    <to>{DateTime}</from>
    <classification>{String}</classification>

    <subclassification>{String}</subclassification>
    <signatureID>{String}</signatureID>
    <computer>{String}</computer>
    <directory>{String}</directory>
    <filename>{String}</filename>
</filter>
```

**xmlRestoreOptions** – options to control how restore is performed

```
<options>
    <allFileVersions>{Boolean}</allFileVersions>
    <overWriteFiles>{Boolean}</overWriteFiles>

    <keepDirStructure>{Boolean}</keepDirStructure>
    <restorePath>{String}</restorePath>
    <username>{String}</username>
    <password>{String}</password>
</options>
```

**restoreID** – ID of restore request. This ID can be used in subsequent calls to retrieve the status of the request.

**xmlResult** – result of the call

```
<result>
    <errorMessage>{String}</errorMessage>
    <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** RestoreFilesByList  
**Description:** Restores a set of files by list.  
**Syntax:** `bool RestoreFilesByList(string xmlFiles, string xmlRestoreOptions, ref string restoreID, ref string xmlResult)`

**Arguments:****xmlFiles** – a list of signature IDs

```
<filter>
  <signatureID>{String}</signatureID>
  ...
</filter>
```

**xmlRestoreOptions** – options to control how restore is performed

```
<options>
  <allFileVersions>{Boolean}</allFileVersions>
  <overwriteFiles>{Boolean}</overwriteFiles>

  <keepDirStructure>{Boolean}</keepDirStructure>
  <restorePath>{String}</restorePath>
  <username>{String}</username>
  <password>{String}</password>
</options>
```

**restoreID** – ID of restore request. This ID can be used in subsequent calls to retrieve the status of the request.**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.



**Method:** SetBlockDispositionStatusByFilter

**Description:** Block or unblock disposition of a set of files by filter.

**Syntax:** bool SetBlockDispositionStatusByFilter(string xmlFilter, bool blockDisposition, ref string xmlResult)

**Arguments:**

**xmlFilter** – the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <from>{DateTime}</from>
    <to>{DateTime}</from>
    <classification>{String}</classification>

    <subclassification>{String}</subclassification>
    <signatureID>{String}</signatureID>
    <computer>{String}</computer>
    <directory>{String}</directory>
    <filename>{String}</filename>
</filter>
```

**blockDisposition** – block or unblock disposition

**xmlResult** – result of the call

```
<result>
    <errorMessage>{String}</errorMessage>
    <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** SetBlockDispositionStatusByList  
**Description:** Block or unblock disposition of a list of files.  
**Syntax:** `bool SetBlockDispositionStatusByList(string xmlFiles, bool blockDisposition, ref string xmlResult)`

**Arguments:**

**xmlFiles** - a list of signature IDs

```
<filter>
  <signatureID>{String}</signatureID>    ...
</filter>
```

**blockDisposition** – block or unblock disposition

**xmlResult** – result of the call

```
<result>
  <errorMessage>{String}</errorMessage>
  <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** SetReadAccessByFilter  
**Description:** Allow or disallow read access to a set of files by filter.  
**Syntax:** bool SetReadAccessByFilter(string xmlFilter, bool allowRead, ref string xmlResult)

**Arguments:** **xmlFilter** – the filter criteria. Wildcards are allowed in all fields except the dates and the signature ID. If the signature ID is provided, all other fields are ignored.

```
<filter>
    <fsorganization>{String}</fsorganization>
    <fsid>{String}</fsid>
    <from>{DateTime}</from>
    <to>{DateTime}</from>
    <classification>{String}</classification>

    <subclassification>{String}</subclassification>
    <signatureID>{String}</signatureID>
    <computer>{String}</computer>
    <directory>{String}</directory>
    <filename>{String}</filename>
</filter>
```

**allowRead** – enable or disable read access

**xmlResult** – result of the call

```
<result>
    <errorMessage>{String}</errorMessage>
    <errorCode>{Int}</errorCode>
</result>
```

An error code of 0 signifies a successful completion.

**Method:** SetReadAccessByList  
**Description:** Allow or disallow read access to a list of files.  
**Syntax:** `bool SetReadAccessByList(string xmlFiles, bool allowRead, ref string xmlResult)`

**Arguments:**

**xmlFiles** - a list of signature IDs

```
<filter>  
  <signatureID>{String}</signatureID>  
  ...  
</filter>
```

**allowRead** – enable or disable read access

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Method:** UploadClientConfiguration  
**Description:** Uploads a client FSW configuration files to the server  
**Syntax:** bool UploadClientConfiguration(string computer, string xmlWatch, ref string xmlResult)

**Arguments:**

**computer** – name of computer hosting the configuration including the domain (domain\computer)

**xmlWatch** – the details of the watch (fswconfig.xml)

**xmlResult** – result of the call

```
<result>  
  <errorMessage>{String}</errorMessage>  
  <errorCode>{Int}</errorCode>  
</result>
```

An error code of 0 signifies a successful completion.

**Note:** Be sure to delete from the fswConfig.xml all data that doesn't belong to the user watches.

## Appendix – Advanced - Configuration Files

A sample client configuration file, containing the list of files to be processed. Note that the watch name must be “api”.

```
<?xml version="1.0" encoding="utf-8"?>
<AssureonDirectoryWatch xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
ILMBuildNo="4.00.0002">
  <globalSettings ID="global" rev="1.0">
    <includeSubDirectories>true</includeSubDirectories>
    <fswEnable>true</fswEnable>
    <syncEnable>false</syncEnable>
    <watchBufferSize>40960</watchBufferSize>
    <processFileOnLastWrite>true</processFileOnLastWrite>
    <processFileOnCreationTime>true</processFileOnCreationTime>
    <processFileOnFileClose>true</processFileOnFileClose>
    <processFileOnDelete>false</processFileOnDelete>
    <processFileOnRename>true</processFileOnRename>
    <delayBeforeProcessing>2000</delayBeforeProcessing>
    <maximumBatchSize>20</maximumBatchSize>
    <storeTimeStampsZipped>false</storeTimeStampsZipped>
    <zippedFileExtension>.AEZ</zippedFileExtension>
    <signFileBeforeSending>false</signFileBeforeSending>
    <signFileCertSN />
    <fileWatchCritical>false</fileWatchCritical>
    <storeOriginalFile>false</storeOriginalFile>
    <storeOriginalFileWithTimeStamp>false</storeOriginalFileWithTimeStamp>
    <storeTimeStampOnLocalMachine>false</storeTimeStampOnLocalMachine>
    <storeTimeStampLocalToFile>false</storeTimeStampLocalToFile>
    <storeTimeStampDirectOrRelative>R</storeTimeStampDirectOrRelative>
    <storeTimeStampDirectory />
    <storeTimeStampRelativeDirectory />
    <recentFileCache>512</recentFileCache>
    <recentFileCacheMax>1024</recentFileCacheMax>
    <recentFileCacheGC>60000</recentFileCacheGC>
    <recentFileCacheEnabled>true</recentFileCacheEnabled>
    <recentFileCachePersist>true</recentFileCachePersist>
    <markTimeStampReadOnly>false</markTimeStampReadOnly>
    <markFileReadOnly>false</markFileReadOnly>
    <logTimeStampsLocally>true</logTimeStampsLocally>
    <directoryPolicy dPID="0" dPName="Policy 0">
      <directoryMatch />
      <fileTypeRule fTRID="0" fName="Rule 0">
        <customerLicense>CLASS00001.SUBCLASS01</customerLicense>
        <removeFileAfterStore>false</removeFileAfterStore>
        <replaceFileWithShortCutAfterTime>
          <enable>true</enable>
          <daysFromDateAccessed>0</daysFromDateAccessed>
          <daysFromDateModified>0</daysFromDateModified>
        </replaceFileWithShortCutAfterTime>
        <include />
        <processFileAfterTime>
          <enable>false</enable>
          <minutesSinceLastAccess>0</minutesSinceLastAccess>
          <minutesSinceModification>0</minutesSinceModification>
        </processFileAfterTime>
      </fileTypeRule>
    </directoryPolicy>
  </globalSettings>
</AssureonDirectoryWatch>
```

## ASSUREON API

```
<minutesSinceCreation>0</minutesSinceCreation>
</processFileAfterTime>
</fileTypeRule>
<fileTypeRule fTRID="1" fName="Rule 1">
  <exclude>
    <filesStartingWith>~</filesStartingWith>
    <extensions>.tmp,.temp,.url,.desktop</extensions>
  </exclude>
</fileTypeRule>
</directoryPolicy>
</globalSettings>
<watchName ID="api" rev="1">
  <directory>c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X</directory>
  <includeSubDirectories>>true</includeSubDirectories>
  <fileWatchCritical>>false</fileWatchCritical>
  <processFileOnLastWrite>>true</processFileOnLastWrite>
  <processFileOnCreationTime>>true</processFileOnCreationTime>
  <processFileOnFileClose>>true</processFileOnFileClose>
  <processFileOnDelete>>false</processFileOnDelete>
  <processFileOnRename>>true</processFileOnRename>
  <delayBeforeProcessing>2000</delayBeforeProcessing>
  <maximumBatchSize>20</maximumBatchSize>
  <recentFileCache>512</recentFileCache>
  <recentFileCacheMax>1024</recentFileCacheMax>
  <recentFileCacheEnabled>>true</recentFileCacheEnabled>
  <recentFileCacheGC>60000</recentFileCacheGC>
  <recentFileCachePersist>>true</recentFileCachePersist>
  <signFileBeforeSending>>false</signFileBeforeSending>
  <signFileCertSN />
  <storeOriginalFile>>false</storeOriginalFile>
  <storeOriginalFileWithTimeStamp>>false</storeOriginalFileWithTimeStamp>
  <storeTimeStampOnLocalMachine>>false</storeTimeStampOnLocalMachine>
  <storeTimeStampLocalToFile>>false</storeTimeStampLocalToFile>
  <storeTimeStampDirectOrRelative>R</storeTimeStampDirectOrRelative>
  <storeTimeStampDirectory />
  <storeTimeStampRelativeDirectory />
  <storeTimeStampsZipped>>false</storeTimeStampsZipped>
  <zippedFileExtension>.AEZ</zippedFileExtension>
  <logTimeStampsLocally>>true</logTimeStampsLocally>
  <removeFileAfterStore>>false</removeFileAfterStore>
  <replaceFileWithShortCut>>true</replaceFileWithShortCut>
  <fswEnable>>true</fswEnable>
  <syncEnable>>true</syncEnable>
  <directoryPolicy dPID="0" dPName="Policy 0">
    <fileTypeRule fTRID="0" fName="Rule 0">
      <customerLicense>CLASS00001.SUBCLASS01</customerLicense>
      <retentionId>DynamicC</retentionId>
      <replaceFileWithShortCutAfterTime>
        <enable>true</enable>
        <daysFromDateAccessed>0</daysFromDateAccessed>
        <daysFromDateModified>0</daysFromDateModified>
      </replaceFileWithShortCutAfterTime>
      <include>
        <file retentionDate="2006-07-17T15:24:27.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44B5142C.DEVDOMAIN.DEVPO.100.1387239.1.3B1E.1.xml</file>
```

## ASSUREON API

```
<file retentionDate="2005-06-13T11:00:00.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Calendar\44B5144B.DEVDOMAIN.DEVPO.100.1387239.1.3B21.1.xml</file>
<file retentionDate="2006-07-18T13:38:25.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\att\44B64CD1.DEVDOMAIN.DEVPO.100.1387239.1.3B24.1\Scheduler.trc</file>
<file retentionDate="2006-07-18T13:38:25.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\att\44B64CD1.DEVDOMAIN.DEVPO.100.1387239.1.3B24.1\GWSyncJob_Allen
Dixon.trc</file>
<file retentionDate="2006-07-18T13:38:25.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\att\44B64CD1.DEVDOMAIN.DEVPO.100.1387239.1.3B24.1\NetMSMQ.trc</file>
<file retentionDate="2006-07-18T13:38:25.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44B64CD1.DEVDOMAIN.DEVPO.100.1387239.1.3B24.1.xml</file>
<file retentionDate="2006-07-26T14:28:32.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44C0E490.DEVDOMAIN.DEVPO.100.1387239.1.3B28.1.xml</file>
<file retentionDate="2006-07-29T08:19:45.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32 2X\xml\Work In
Progress\44C482A2.DEVDOMAIN.DEVPO.100.1387239.1.3B2B.1.xml</file>
<file retentionDate="2006-07-29T08:35:34.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32 2X\xml\Work In
Progress\44C48656.DEVDOMAIN.DEVPO.100.1387239.1.3B2C.1.xml</file>
<file retentionDate="2006-07-29T11:08:09.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32 2X\xml\Work In
Progress\44C4AA19.DEVDOMAIN.DEVPO.100.1387239.1.3B2D.1.xml</file>
<file retentionDate="2006-07-30T12:45:19.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44C61260.DEVDOMAIN.DEVPO.100.1387239.1.3B32.1.xml</file>
<file retentionDate="2006-07-31T16:47:00.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Calendar\44C79C9E.DEVDOMAIN.DEVPO.100.1387239.1.3B35.1.xml</file>
<file retentionDate="2006-07-31T16:47:00.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Calendar\44C79C9E.DEVDOMAIN.DEVPO.100.1387239.1.3B36.1.xml</file>
<file retentionDate="2006-07-31T16:47:50.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Calendar\44C79CB6.DEVDOMAIN.DEVPO.100.1387239.1.3B38.1.xml</file>
<file retentionDate="2006-08-19T11:59:59.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Calendar\44C79CB6.DEVDOMAIN.DEVPO.100.1387239.1.3B39.1.xml</file>
<file retentionDate="2006-07-31T16:48:30.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44C79CDE.DEVDOMAIN.DEVPO.100.1387239.1.3B3B.1.xml</file>
<file retentionDate="2006-08-07T09:15:58.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32 2X\xml\Work In
Progress\44D06D4E.DEVDOMAIN.DEVPO.100.1387239.1.3B40.1.xml</file>
<file retentionDate="2006-08-07T11:00:00.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Calendar\44D06E38.DEVDOMAIN.DEVPO.100.1387239.1.3B42.1.xml</file>
<file retentionDate="2006-08-07T13:30:00.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Calendar\44D06EF3.DEVDOMAIN.DEVPO.100.1387239.1.3B44.1.xml</file>
```



## ASSUREON API

```
<file retentionDate="2006-08-07T13:30:00.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Calendar\44D06EF3.DEVDOMAIN.DEVPO.100.1387239.1.3B45.1.xml</file>
  <file retentionDate="2006-08-08T13:08:42.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44D1F55A.DEVDOMAIN.DEVPO.100.1387239.1.3B4C.1.xml</file>
    <file retentionDate="2006-08-08T13:08:42.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44D1F55A.DEVDOMAIN.DEVPO.100.1387239.1.3B4D.1.xml</file>
      <file retentionDate="2006-08-16T09:28:27.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44DC4DBB.DEVDOMAIN.DEVPO.100.1387239.1.3B58.1.xml</file>
        <file retentionDate="2006-08-19T14:30:52.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44E0891C.DEVDOMAIN.DEVPO.100.1387239.1.3B5B.1.xml</file>
          <file retentionDate="2006-08-19T14:30:52.0000000-
04:00">c:\EmailArchive\ArchivesNexsan\Agent32 1X Agent32
2X\xml\Mailbox\44E0891C.DEVDOMAIN.DEVPO.100.1387239.1.3B5C.1.xml</file>
            </include>
              <processFileAfterTime>
                <enable>>false</enable>
                <minutesSinceLastAccess>0</minutesSinceLastAccess>
                <minutesSinceModification>0</minutesSinceModification>
                <minutesSinceCreation>0</minutesSinceCreation>
              </processFileAfterTime>
            </fileTypeRule>
          </directoryPolicy>
        <markFileReadOnly>>false</markFileReadOnly>
        <markTimeStampReadOnly>>false</markTimeStampReadOnly>
        <watchBufferSize>40960</watchBufferSize>
      </watchName>
    </AssureonDirectoryWatch>
```



**Nexsan Headquarters**

325 E. Hillcrest Drive, Suite #150  
Thousand Oaks, CA 91360 USA

**Nexsan Shipping**

302 Enterprise Street , Suite A  
Escondido, CA 92029 USA

**Nexsan Canada**

1405 Trans-Canada Highway, Suite 300  
Dorval, QC H9P 2V9 Canada

**Nexsan UK**

Units 33–35, Parker Centre, Mansfield Road  
Derby, DE21 4SZ United Kingdom

Copyright © 2010—2019 Nexsan Technologies, Inc.. All rights reserved.  
Nexsan® is a trademark or registered trademark of Nexsan Technologies, Inc..  
The Nexsan logo is a registered trademark of Nexsan Technologies, Inc..  
All other trademarks and registered trademarks are the property of their respective owners.

This product is protected by one or more of the following patents, and other pending patent applications worldwide:  
United States patents US8,191,841, US8,120,922;  
United Kingdom patents GB2466535B, GB2467622B, GB2467404B, GB2296798B, GB2297636B