Novell

SMS and Backup Application Development

www.novell.com

QUICK START

SMS Recommendations

Introduction

Open Enterprise Server (OES) offers customers the flexibility to deploy services on NetWare® or SUSE™ with common management experiences on both platforms. Open Enterprise Server consists of the following software components:

- NetWare 6.5, Support Pack 3
- SUSE LINUX Server 9, Enterprise Edition
- Nterprise Linux Services 2.0

In order to provide a consistent backup interface, Storage Management Services (SMS) API framework is being made available on SUSE LINUX Enteprise Server (SLES). With the OES release, SMS has a single consistent interface to backup file systems on NetWare, OES Linux (including NSS on SLES) and other Novell applications. The API set has also been enhanced to include newer functionality.

This document provides an overview of the functionality, recommendations for their usage and limitations and is expected to help backup application development. The primary focus is to enable backup application developers to exploit these services in a consistent manner on OES NetWare and OES Linux.

PURPOSE

The primary objectives of this document are:

- To provide a basis for planning and implementation to support backup of the NSS file system on OES Linux.
- To provide, in a consolidated form, details of newer features that have been introduced in the past couple of years and how they can be better exploited.

AUDIENCE

The target audience for this document are backup application developers and corporations that already have experience with SMS in building applications for NetWare. The document is expected

Novell®

to serve as a guide to understand newer features and functionality that have been recently implemented.

SCOPE

Focus Of the Document

- This main objective of this document is to provide an overview of recently introduced features and functionality only. It will not focus on existing functionality already implemented by many commercial backup applications.
- The focus of this document is to provide recommendations on how newer API can be better exploited. It will also bring out current limitations.

What the Document is Not

- This document will not describe API in detail, and will instead, refer to the SMS API documentation (see “References” on page 2).
- This document is NOT a tutorial on SMS and all its features, or how to go about developing a backup application.
- The document will also not duplicate information already available in other locations. Instead, it will refer to documentation where ever necessary.
- This document focuses on file system TSA features on OES NetWare and OES Linux. It does not focus on TSAs for applications such as GroupWise. However, as it can be noted, all TSAs will comply with the API definition present in this version of the API set.

References

This document is intended to be read in conjunction with the following:

- **SMS API documentation:** The SMS Developer Components Documentation (http://developer.novell.com/ndk/doc/smscomp/sms_docs/data/h6nkslcr.html) has been updated with the latest information and ships with this developer release.

  Every subsection in this document has an “Related documentation” section that refers this document and sections within it.
- **Sample Code:** A backup application sample (DE) is provided with this release for both NetWare and OES Linux. This sample code can be used as a reference to understand the usage of SMS API.
- **DE Documentation:** The demoengine.html provides information on DE usage and also a step by step detail on what API is exercised in that operation. This document provides pointers on how the sample code can be read.

SMS: Components and Features

This section provides a brief introduction to SMS, its components and provides a guide to read the rest of the document. Although this document is intended for developers who are already familiar with SMS, this section provides a brief survey to make references in the rest of the document more meaningful.

SMS

Storage Management Services (SMS) is an API framework that backup applications consume in order to provide a complete backup solution on NetWare and Open Enterprise Server (OES Linux). SMS exposes a single consistent interface across all file systems and applications. Thus, backup applications written to this API can work with NSS, GroupWise™ or Novell iFolder®. The SMS framework is implemented by two main components - The Storage Management Data Requestor (SMDR) and the Target Service Agent (TSA). SMDR defines the API framework and a TSA provides an implementation of the API for a particular target.

TSA

The Target Service Agent (TSA) is a component that provides abstractions of a particular backup target. The TSA typically resides on the server where the target is located and uses file system or application interfaces (based on the target) to extract data and transforms it to a continuous stream of data objects. This stream is usually represented in a standard format such as SIDF. TSAs exist for the file system, GroupWise and eDirectory™. TSAs for other targets are likely to be built in the future.

SMDR

SMDR is the communication agent between the backup application and the target service agent and it defines the API framework. SMDR deals with the movement of data from one server to another while providing location transparency. It provides the communication channel between the servers involved and abstracts details of the protocols and network abstractions used in the communication.

Consistent API Framework

SMS has traditionally been available on NetWare. With the OES release, SMS is available on both NetWare as well as OES Linux and also carries the same design philosophy of having a single consistent API framework for applications (such as iFolder, GroupWise) or file systems (NSS, VFS-compliant) across different platforms. This will help backup application development as it would have a single view of all Novell backup targets.

Why do we need SMS on OES Linux?

OES Linux already has several applications for backup such as TAR, CPIO, etc. SMS on OES Linux is intended to add value to the existing collection by providing a consistent API for file systems and applications. In future, SMS may have the ability to provide different formatters as well which is likely to include TAR, and the current PAX standard, apart from SIDF.

SMS also brings to OES a very important ability to be able to backup and restore NSS file system data. Although NSS will be exposed as a VFS-compliant file system, the VFS interfaces are inadequate to be able to backup NSS file system attributes and data like rich ACL model, trustees, multiple data streams. SMS will expose all NSS data through the API ensuring complete backup protection of the file system on OES Linux for enterprises.

New SMS Features

SMS has, over the last couple of years, implemented and/or exposed several new features that can be used by backup applications. These include functionality such as extended character passwords, Unicode streams, performance enhancements and more recently, its availability on OES Linux.

In this document, the focus is on these new features that can be exploited by backup applications to enhance the options provided to the customer today. All features are available consistently on OES NetWare and OES Linux .

NOTE: None of the features discussed in the subsequent sections have dependencies on each other. Each feature can be independently implemented. They have been described in this document in no particular order although cross-platform support for SMS and hence from backup applications would be the most important need for customers today.

Cross Platform SMS

This section would focus on changes that would be required in backup applications to work seamlessly with SMS from the OES release. For a detailed description of new binaries and their usage on OES Linux, refer “Appendix A - SMS binaries on OES Linux” on page 25. For steps to install and verify if SMS is working, refer to SMS Readme (http://forgeftp.novell.com/smscomp/README.html).

SMS IMPLEMENTATION CONCEPTS

This section describes the implementation of the SMS components (TSA and SMDR) on OES Linux and how they behave differently from their implementations on NetWare. Some information in this section may not be required for coding and development of the backup application itself, but will be needed in order to troubleshoot problems that are likely to arise during such a development.

Discovery and advertisement

Discovery and advertisement of SMDRs in the network enables the backup application to locate targets running the SMDR service.

On Netware, the discovery and advertisement mechanisms used are SLP and SAP. SLP is widely used since it is based on TCP/IP, while SAP is based on SPX/IPX protocol. DNS/hosts resolution is also supported.

On OES Linux, the discovery and advertisement mechanism used is SLP. Currently, OpenSLP libraries are used for both discovery and advertisement. For advertisement, it is mandatory that any SLP (protocol version 1 or greater) compliant service agent be running on the server where SMDR runs. DNS/hosts resolution is also supported.

Resources

In SMS, the primary resources are those that are the highest level entities under a target service. The secondary resources can be viewed as the children of the primary resources. On NetWare file system target services, examples of primary resources are volumes, File Server, Server Specific Info, etc. On OES Linux file system target services, all mount points are considered as a primary resource. Directories and files under the primary resources are referred to as secondary resources.

SMS follows the rules listed below on OES Linux file system target service for browsing resources:

- Resources are defined to be mutually exclusive, hence mount points within mount points are excluded while browsing the parent mount point.
- Resources exclude pseudo-file systems like proc, tmpfs, _admin, devpts, usbfs etc. They also exclude devices that fall under the /dev directory.
- Resources that can be viewed are limited to the access rights of the current connection. If a mount path is determined to be under a path, that the current connection does not have access to, then such primary resources are excluded form the connections list.

NameSpaces

NetWare can support up to four name spaces on any given primary resource. On OES Linux each primary resource contains only one name space that is supported natively by the platform.

The name space supported on OES Linux follows the following broad guidelines,

- Names are case sensitive
- Names contain only the “/” separator between each node in the name

The NFS name space is defined based on the same rules as above on NetWare. Hence, on OES Linux only the NFS name space is used to represent all names returned from the SMS interface.

Each name space on NetWare also defines the name space separators that are associated with them. There are two separators used in NetWare. The first separator is used to distinguish between a primary resource and the following secondary resource. The second separator separates each secondary resource names as found in the respective targets hierarchical structure. On OES Linux the primary resource is not distinguished from the secondary resource. Hence only the second separator is used distinguish between resource nodes. This differs from the NFS name space as defined and processed on NetWare.

NOTE: : Internally when backing up NSS file system on OES Linux, all name spaces as supported by NSS would be represented in SIDF and backed up. Only the NFS name space as defined above would be returned in the dataSetNames parameter from the NWSMTSScanDataSetBegin, NWSMTScanNextDataSet and NWSMTSScanDataSetContinue functions.

Connection Functions

The file system TSAs target service name is defined to be “<server name>.NetWare File System” on NetWare and “<server name>. Linux File System” on OES Linux. To work with the file system TSA the mentioned TSA name as returned by the API needs to be used.

On NetWare, the credentials used are the eDirectory login credentials. On OES Linux, the file system target service uses the credentials of an user having an identity on the OES Linux system. After validating the credentials, the respective users' user ID and group ID is used for access control.

The file system TSA also supports PAM based authentication on OES Linux. Hence, it is possible to use other password based mechanisms to authenticate to the TSA. With OES, Linux User Management(LUM) provides PAM mechanisms to enable eDirectory based authentication of LUM users. For more information see Novell Linux User Mangement 2.2 Technology Guide (http://www.novell.com/documentation/oes/index.html?page=/documentation/oes/lumadgd/data/front.html#bktitle).

Backup Functions

NWSMTSScanDataSetBegin starts the scan of a resource and returns information pertaining to the first data set encountered. NWSMTSScanNextDataSet continues a scan started by the NWSMTSScanDataSetBegin API. Both APIs return name information (dataSetNames) and meta data information (scanInformation) of the data set being scanned.

For the file system target service on OES Linux, the dataSetNames contain only the NFS name space name. The scanInformation contains meta data information that is populated from the stat system call as given below. The table below only gives mappings between VFS-compliant file system returned meta data to the structure elements in scanInformation. Any element not mentioned in the list below contains data as before.

scanInformation Element	stat Information
attributes	Only gives NWFA_DIRECTORY information
creatorID	**Not** populated
modifiedFlag	**Not** populated
accessDateAndTime	Filled with the stat.st_atime in DOS time format
createDateAndTime	**Not** populated
modifiedDateAndTime	Populated with the stat.st_mtime in DOS time format
archivedDateAndTime	**Not** populated

Additionally the otherInformation field of the NWSM_SCAN_INFORMATION structure has been enhanced to hold extensions. For VFS-compliant file systems additional meta data is provided in this field as a NWSM_SCAN_INFO_EXTN_NFS_TAG extension that contains the

NWSM_SCAN_INFO_EXTN_NFS_DATA_1 structure. This can be used to access additional meta data that is VFS specific and not represented by the NWSM_SCAN_INFORMATION fields.

NOTE: For NSS file system on OES Linux, the fields (that are not populated as in the table above) would be populated with the appropriate NSS information.

Option Functions

The option functions that return open mode and scan types are currently supported at par on OES Linux as compared to NetWare. Although most of the open mode or scan type options have no corresponding meaning for VFS-compliant file systems, these would be honored when backing up NSS file systems under OES Linux and ignored for other VFS-compliant file systems.

The following tables list the supported options on non NSS file systems on OES Linux.

Table 1 *NWSMTSGetOpenModeOptionString*

optionNumber parameter	Description
0	Do not read/write the data sets data streams

Table 2 *NWSMTSGetTargetScanTypeString*

typeNumber parameter	Description
0	Do not traverse the file system tree
12	Do not read a data sets data stream.

The NWSMTSGetTargetResourceInfo API returns information pertaining to primary resources on the target service. On OES Linux, the parameters as listed in the following table are populated, parameters that are not populated are zeroed. These parameters are populated using the return values of the statfs call.

Table 3 *NWSMTSGetTargetResourceInfo*

NWSMTSGetTargetResourceInfo Parameter	statfs Information
totalBlocks	f_blocks
freeBlocks	f_bfree
blockSize	f_bsize

Additional information regarding primary resources are returned by the NWSMTSGetTargetResourceInfoEx API as a NWSM_RESOURCE_INFO_EXTN_UNIX_TAG extension that contains the NWSM_RESOURCE_INFO_EXTN_UNIX_DATA_1 structure. This can be used to determine various mount options used and the mount type.

NWSMTSGetUnsupportedOptions lists the set of unsupported options that the TSA does not support. Between primary resources in OES Linux it is possible that the resource itself does not support certain additional TSA features, for example modify bits on files. The API NWSMTSGetTargetResourceInfoEx returns the NWSM_RESOURCE_INFO_EXTN_UNSUPPORTED_TAG extension that contains the NWSM_RESOURCE_INFO_EXTN_UNSUPPORTED_DATA_1 structure. This additionally provides the set of TSA options that the particular resource does not support.

Miscellaneous Functions

NWSMTSSetArchiveStatus is used to set the archive dates and attribute bits for data sets. On OES Linux the archive dates and bits are not supported on non-NSS file systems. A cross-platform and file system agnostic determination of archive support on a resource needs to be determined.

As a result of non availability of archive dates and attributes for non NSS based file systems on OES Linux, incremental/differential backups supported based on the NWSM_EXCLUDE_ARCHIVED_CHILDREN bit set on *scanType* parameter of the NWSM_SCAN_CONTROL structure is not possible.

The NWSMTSGetTargetResourceInfoEx API can be used to retrieve the unsupported TSA options by a particular resource to determine support for NWSM_BACK_MODIFY_FLAG and NWSM_BACK_ARCHIVE_DATE_TIME. Based on the support of these bits incremental and differential backups using the scanType bits can be performed.

Utility Library

With previous SMS NDK releases, the sms utility library was shipped as static library (sms.lib). Backup applications needed to link statically with this library and required a relink if there were any updates to the library. In order to avoid this, it will be shipped as an NLM (smsut.nlm) on NetWare in OES. This NLM can be loaded independently and can also be updated without changes to backup applications. On OES Linux, the utility library is a shared object (libsmsut.so), which can be loaded at runtime. The utility library will be installed along with SMS installation.

The utility library has been enhanced to provide extension API that can be used to retrieve extension information encoded in the various SMS APIs. In this release of the NDK the NWSM_SCAN_INFORMATION and the NWSMTSGetTargetResourceInfoEx provide extension information that can be passed to these extension APIs to retrieve relevant extension structures.

Suggested Changes for Backup Applications

- Any cross-platform solution must use the NWSMTSListSupportedNameSpaces API to determine support for the name spaces used by the target service before passing in names to the target service in various name spaces.
- Any cross-platform application using SMS would need to call NWSMTSGetNameSpaceTypeInfo to get the first and second separators for name spaces used by the application. This would ensure that proper first and second separators are used on different platforms for the same name space.
- The TSA name returned by NWSMListTSAs needs to be used. On NetWare, the file system TSAs target service name is defined to be “<server name>.NetWare File System”. On OES Linux the file system TSAs service name is defined as “<server name>.Linux File System”.
- Backup applications need to understand that for the file system target service on OES Linux, the dataSetNames contain only the NFS name space name. The backup application will also need to recognize the interpretation of scanInformation for VFS-compliant file systems as described in “Backup Functions” on page 7.
- Backup applications need to recognize that all open modes and scan types continue to be supported although some of them will not have an interpretation on non-NSS file systems. This has been done in order to maintain the consistency of the API. For details, refer “Option Functions” on page 8.
- Backup applications that require VFS specific primary resource attributes can use the NWSMTSGetTargetResourceInfoEx function retrieve additional meta data concerning resources. For details refer “Option Functions” on page 8 and “Miscellaneous Functions” on page 9.
- Backup applications will not be able to support incremental/differential backups based on the NWSM_EXCLUDE_ARCHIVED_CHILDREN bit for non-NSS file systems. NWSMTSGetTargetResourceInfoEx API can be used to determine support for this bit by a particular primary resource.
- Backup applications will need to now link with the SMSUT dynamic library on both NetWare and OES Linux. On NetWare, SMSUT.IMP needs to be included in the import files and SMSUT.NLM added to the auto load list. On OES Linux, the backup application modules needs to link with libsmsut.so.

NetWare Emulation mode on OES Linux

This solution is provided for backup applications vendors to enable the backup of Linux NSS file system as NetWare file system using their existing back up software.

Linux file system TSA works in three modes; Linux, NetWare, and Dual. Linux mode being the default mode.

- **Linux mode**

  In this mode, file system TSA exposes “<server name>.Linux File System” as a target service and all Linux mount points are exposed as resources.

- **NetWare mode**

  In this mode file system TSA exposes “<server name>.NetWare File System” as a target service and the mounted NSS volumes are exposed as resources.

- **Dual Mode**

  In this mode, TSAFS exposes “Linux File System” and “NetWare File System” as target services and resources will be dependent on the Target Service.

Emulation mode virtualises Linux TSAFS as a NetWare TSAFS, when used in the NetWare mode. Hence, moving data from OES NetWare to OES Linux is treated analogous to moving data between two NetWare servers. Existing backup applications for NetWare can extend their backup support for Linux treating the Linux Server as a NetWare Server. Linux and NetWare modes are inter-operable. Backup data with one mode can be restored using another mode.

For complete information on configuring TSA to use dual mode, refer to the NetWare Emulation Mode on OES section in the *Using SMS chapter of the Storage Management Services Administration Guide* (http://www.novell.com/documentation/oes/smsadmin/data/hhc3nq5m.html#bv8st4y).

LIMITATIONS

This section lists the limitations of the binaries in this release. It is expected that these limitations will not hinder backup application development.

- *Passwd* and *shadow* passwords are used by the file system target service to validate credentials. Access to the password files on OES Linux require that the process accessing them run as root. Hence, SMDR and file system target service will run with root on OES Linux.
- Target service for the OES Linux file system does not define the two special resources “Linux Server” and “Server Specific Information” as its NetWare counterpart.

RELATED DOCUMENTATION

Following references point to the SMS API documentation that ships along with this developer release available at NDK: SMS Developer Components Documentation (http://

developer.novell.com/ndk/doc/smscomp/index.html?page=/ndk/doc/smscomp/sms_docs/data/h6nkslcr.html).

- Refer the Backup Functions section in the Target Services Functions Chapter of the API Documentation for the following backup related API
  - NWSMTSScanDataSetBegin
  - NWSMTSScanNextDataSet
  - NWSMTSScanDataSetContinue
- Refer the Connection Functions section in the Target Services Functions Chapter of the API Documentation for the following Connection relates API
  - NWSMListTSAs
  - NWSMTSConnectToTargetServiceEx
  - NWSMTSConnectToTargetService
- Refer the Option Functions section in the Target Services Functions Chapter of the API Documentation for the following option related API
  - NWSMTSGetOpenModeOptionString
  - NWSMTSGetNameSpaceTypeInfo
  - NWSMTSGetTargetScanTypeString
  - NWSMTSGetTargetResourceInfo
  - NWSMTSListSupportedNameSpaces
  - NWSMTSGetTargetResourceInfoEx
  - NWSMTSGetUnsupportedOptions
- NWSM_SCAN_CONTROL definition can be found in the Target Services Structures chapter of the API documentation.
- For examples on how to use these API refer to the demonstration engine documentation and code.
- Refer the Utility Library Concepts chapter of the API Documentation for the following extension related API
  - NWSMGetExtension
  - NWSMGetFirstExtension
- NWSM_SCAN_INFORMATION definition can be found in the Utility Library Concepts chapter of the API documentation.

- NWSM_RESOURCE_INFO_EXTN_UNIX_DATA_1 definition can be found in the Utility Library Structures chapter of the API documentation.
- NWSM_RESOURCE_INFO_EXTN_UNSUPPORTED_DATA_1 definition can be found in the Utility Library Structures chapter of the API documentation.
- NWSM_SCAN_INFO_EXTN_NFS_DATA_1 definition can be found in the Utility Library Structures chapter of the API documentation.

Performance Model

Background

Backup has traditionally been serial in nature. While this approach was acceptable in the past, it is certainly not effective as today's backup involves several terabytes of data and tape devices are able to service several megabytes of data per second. In addition to this, file systems are generally optimized for servicing multiple client requests simultaneously.

In order to take advantage of predictable access of files during backup, TSAFS was re-designed to incorporate read-ahead caching and the new version shipped first in NetWare 6.5 (Subsequently, TSAFS was also made available on NetWare 6.0 SP 4 as well). With the new model, performance improved dramatically. However, the manner of backup applications' usage of SMS API does impact the extent to which this benefit is transferred to the end-customer. This section contains some of these recommendations.

Performance enhancements

TSAFS was redesigned to deliver performance without changing the API set definition in order to provide backward compatibility with existing backup application implementations. This was accomplished by de-coupling the serial usage of the file system interface in TSA library. In this model, the TSA takes advantage of the predictable nature of requests and caches data ahead of time, so that backup application requests can be serviced from the memory instead of the disk.

With these enhancements, TSAFS delivers dramatic increase in performance. As with any system, peak performance is derived when the system is correctly tuned. This involves tuning the TSA parameters to ensure maximum throughput. More importantly, backup applications can derive better performance by recommended usage of the SMS API. The focus of the next section is to provide specific recommendations that will help improve overall backup throughput while using TSAFS.

Suggested changes to backup applications

- Backup applications must adhere to recommended calling sequence as described in the chapter on *Performance and File system TSA* in the SMS API documentation. Example implementation is also provided as a sample (TSATEST) in the developer kit.

- Backup applications that call NWSMTSScanDataSetBegin on every dataset can be modified to call NWSMTSScanDataSetBegin on the primary resource followed by NWSMTSScanNextDataSet for data sets under this resource.

- Backup applications that call NWSMTSScanDataSetBegin for every directory should also avoid doing so. Instead the backup application should call NWSMTSScanDataSetBegin for

the primary resource followed by NWSMTSScanNextDataSet for all other data sets under this resource.

- Backup applications should provide the highest level resource to be backed up in order to maximize the advantage of the caching model incorporated within the TSA. (Providing a single data set at a time (in point 2 above) provided the lowest level resource to back up thereby limiting any optimization that the TSAFS can do.)

- Implementing selection lists in the backup application reduces the effectiveness of the read-ahead as a subset of the data sets read by TSAFS will be discarded by the backup application. These unnecessary read-aheads impact the average read service time for data sets that are actually backed up. This can be mitigated by either providing the TSA with a filtered list of data sets to be backed up or using the TSA selection list.

  Alternately, if the backup applications needs to filter data sets, the NWSM_USE_CACHING_MODE flag can be set to false using the NWSMTSConfigureTargetService API to prevent a degrade in the performance.

- It is advisable not to change the open mode during the course of the backup, if possible. The open mode provided to TSAFS is used to build a cache of open files. If the open mode is changed, all open and read-ahead operations have to be stopped, the caches destroyed and rebuilt. This will affect the average time to service a data set request.

- In the new model, the open operation triggers a chain of events such as open and read-aheads. Given this, it is recommended that open is used only during the backup and not during a scan for any reason.

- It is recommended that NWSMTSScanDataSetEnd() can be used to terminate the job prematurely. This assumes more importance in the current model. If this API is not used, scan, open and read-ahead continue impacting the overall system performance.

- In this model, NWSMTSReturnToParent() is likely to degrade system performance as all open and read-ahead operations have to be stopped, and the caches should be destroyed and rebuilt. While this is the case, the API is useful in certain circumstances. Hence, the benefits should be understood before its usage.

- It is recommended that Close is called immediately on all data sets that were successfully read. This allows TSAFS to effectively use its resources and also prevent resource starvation in extreme cases.

- It is recommended that backup applications use TSA provided mechanisms for incremental/differential backups. In cases where it is unavoidable and the backup application needs to process the incremental/differential selection of data sets, it is recommended that the backup application set the NWSM_USE_CACHING_MODE to false using the NWSMTSConfigureTargetService API.

Testing for performance

TSATEST is a utility that reads data sets from a target, and then discards them. This, in effect, is equivalent to a backup system that has an infinite capacity and infinite speed tape sub-system. This configuration allows the optimization of the all other sub-systems other than the tape-subsystem. TSATEST can be used for two purposes:

- TSATEST can be used to fine tune the software components to derive the maximum performance rated by the hardware.
- TSATEST can then be used to benchmark against a backup application.

TSATEST is available as part of the developer kit along with its source which can be treated as the recommended implementation for achieving high performance.

Limitations

- When using SMDR for remote backups, due to the serial (RPC-like) interface of SMDR the performance of backups would be slower than local backups.
- TSATEST can only be utilized to determine performance issues of software components like the TSA or the file system. Any bottlenecks due to setting up suboptimal hardware components cannot be analyzed or overcome using optimal software and TSATEST would show poor performance if the underlying hardware is unable to demonstrate a reasonable measure of scalability.

Related Documentation

Following references point to the SMS API documentation that ships along with this developer release.

- Refer the Backup Functions section in the Target Services Functions Chapter of the API Documentation for the following backup related API:
  - NWSMTSScanDataSetBegin
  - NWSMTSScanNextDataSet
  - NWSMTSScanDataSetEnd
- NWSMTSReturnToParent definition can be found in the Target Services Functions Chapter of the API Documentation
- NWSMTSConfigureTargetService definition can be found in the Target Services Functions Chapter of the API Documentation
  - For examples on how to use these API refer to the demonstration engine documentation and code. Also, refer to the TSATEST code provided with this developer release.

Unicode Support

Background

The NSS file system stores path names natively in Unicode. However, prior to NetWare 6.5 SP2, most backup applications, SMS and the NCP clients used MBCS representations for path names internally. This implied that a conversion was required between MBCS and Unicode and vice versa for every data set that was processed by a backup application. These conversions are usually done using the servers' locale as a basis which may not always result in an appropriate conversion. Further, it is possible that even with a single locale, round-trip conversions may cause certain characters to be mis-represented. For example, the Unicode character to represent 0xDA in the MAC code page converts to a "/" separator in MBCS. Given this, the best approach would be to ensure there are no conversions done during backup and restore.

In order to accomplish this, all of Novell's services in NetWare 6.5 SP2, including NCPs and SMS, have been enabled to process UTF-8 representations of data. SMS now stores MBCS as well as UTF-8 representations of the path names in the SIDF stream and provides API to access the UTF-8 representations of path names. Backup applications can now use the SMS API to provide a path from the file system to tape that is free of conversions, which in turn is preserves path names accurately in all environments.

Unicode support in SMS

SMS has been enhanced to support storing UTF-8 representations in addition to MBCS in the SIDF data. SMS interfaces that accept path names or file names have been enhanced to accept UTF-8 representations. This means that operations such as scan, that accept path names, can now accept UTF-8 representations and also return information in UTF-8. Importantly, this enables backup applications to now process the entire backup session (from the target to the tape) using UTF-8.

Unicode support in SMS has been implemented in such a way so as to not change the API call sequence. Instead, the behavior of the API can be controlled by the nameSpaceType parameter. It is important to understand that the nameSpace and the nameSpaceType are different. nameSpaceType refers to the format in which the nameSpace is represented. Currently, the formats supported are MBCS and UTF-8. For example, nameSpaceType for DOS name space is represented as DOSNameSpace to denote MBCS format and DOSNameSpaceUTF8Type for UTF-8 format. An additional API NWSMTSGetSupportedNameTypes, has been added to determine the different name space types that a TSA can support.

NOTE:

This developer release contains a TSAFS that implements and exposes the UTF-8 enabled API on both OES NetWare and OES Linux.

With NetWare 6.5 Support Pack 2, TSAFS includes the functionality to store UTF-8 representations in the SIDF data, but did not have interfaces to expose UTF-8 data. This effort removed conversions from the TSAFS except at the interface with the backup application. It also implemented solutions that intelligently used the UTF-8 representations (where applicable, during restores) that served as a workaround for common customer problems. However, all issues are expected to be fully and effectively resolved when backup applications provide a path of no conversion from disk to tape using the API exposed by SMS.

Suggested changes to backup applications

- Backup applications have a need to display directories and files to the end-user. This needs to be considered independently of SMS and appropriate conversions made as necessary.
- Rename restores take an additional initial path name that is usually supplied by the user. Backup applications would need to convert this to UTF-8 and then construct the new pathname. While storing name lists to tape during backup, store the UTF-8 representation of the same to overcome using MBCS on restores. Also, using a parent-child relationship between data sets, it is possible not to rename every data set but, rename only parents and use the parent handles to restore children.
- Backup applications need to review APIs that accept nameSpaceType as an argument and enhance it to handle UTF-8 name types. The recommended mode of working with a TSA is to use UTF-8 names as returned by the TSA and pass the same to other relevant TSA APIs.
- During selection of data sets to backup or restore, use the UTF-8 representation for the same to pass data to the TSA APIs, this information would be available in all the name lists that are returned by the TSA.
- It is recommended to store both MBCS and UTF-8 representations of the path names on tape in order to support restores to any TSA (even those that do not support UTF-8)

Example: Scanning using a UTF-8 enabled target Service

The following example is used to illustrate how path names can be natively processed in UTF-8 without any conversions in the backup application itself. It presupposes successful connection establishment and detection of the target service's support of UTF-8 name type

1. Use the NWSMTSListTSResources to get a list of supported target service resources. The name space type for resources are always NWSM_TSA_DEFINED_RESOURCE_TYPE, these are represented in MBCS for all target services.
2. While starting a scan using the NWSMTSScanDataSetBegin API, pass in the major resource as received in Step 1. Additionally, specify the return name space types that is required by the backup application, in our current context this would be NWSM_ALL_NAME_SPACES_UTF8. This would return the dataSetNames parameter that contains data set names encoded in UTF-8.

3 For further scanning of non-major resources recursively, use the UTF-8 names as received from the TSA in Step 2 as the resourceName parameter to the next call to NWSMTSScanDataSetBegin.

The above steps listed for scans can be extended for backups as well. During backups the NWSMTSOpenDataSetForBackup/NWSMTSReadDataSet API s only return SIDF encoded data. The SIDF encoding of path names are done as specified in the NWSM_SCAN_CONTROL structure specified in NWSMTSScanDataSetBegin.

NOTE: All interfaces that have a nameSpaceType parameter support the UTF-8 as a name space representation. The above steps would be valid in all such cases.

LIMITATIONS

- As stated earlier, any character conversion between MBCS and UTF-8 is likely to cause certain characters to be mis-represented. While conversions can be avoided in most cases, any instance requiring a conversion would suffer from limitations imposed by the conversion. For example, backing up data from a NSS volume (which stores path names natively in Unicode) and restoring to a NetWare traditional volume (which stores path names natively in MBCS) may cause certain characters to be mis-represented.
- NWSM_DATA_SET_NAME_LIST structure as returned by the TSA is limited in size to bufferSize which can store upto 65536 bytes. If the backup application requests both MBCS and UTF-8 representations of path names, this reduces the total path length of a data set returned by the NWSM_DATA_SET_NAME_LIST at a time by least half.

RELATED DOCUMENTATION

Following references point to the SMS API documentation that ships along with this developer release.

- NWSMTSGetSupportedNameTypes definition can be found in the Target Services Functions Chapter of the API Documentation
- For examples on how to use this API refer to the demonstration engine documentation and code.

Extended Character Passwords

Background

There has been a growing need to support passwords that have characters from beyond the standard ASCII definition. Although, passwords were earlier represented in MBCS, there were several problems. With NetWare 6.5, Novell introduced a feature commonly called Universal password support. One of the related features that has direct relevance to backup is the support of passwords that can contain any character representable in Unicode. With this implementation, all passwords are converted to their UTF-8 representation and processed. This was implemented by changes to both the client and server and hence needed all clients and servers upgraded in order to truly support international passwords in the network. However, it is possible to have a mixed network of old and newer clients or servers if the passwords remain in the standard ASCII range.

Extended character password support in SMS

With NetWare 6.5, SMS provides interfaces with which backup applications can now provide international characters in the password represented in UTF-8. SMS then uses this to authenticate to eDirectory where supported.

Suggested changes to backup applications

- NWSMTSConnectToTargetServiceEx needs to be used in place of NWSMTSConnectToTargetService. This new API supports both UTF-8 representation as well as the older MBCS representation and hence can completely replace its older counterpart (NWSMTSConnectToTargetService).
- It is recommended that UTF-8 representation of passwords be used always. This would work transparently even in older scenarios as the UTF-8 and MBCS representations are equivalent for the ASCII range of characters.

Limitations

- If passwords contained non-ASCII characters and older (pre-NetWare 6.5) clients are used to set the password, usage of NWSMTSConnectToTargetserviceEx would fail. In such situations, the best solution in such cases is to try with the MBCS password itself and set the optionFlag to NWSM_AUTH_RAW_DATA.
- If the target service does not support the UTF-8 password option (as can happen on older TSAs or servers), the call to the API will fail. The backup application would then have to use the MBCS option to authenticate.
- NWSMTSConnectToTargetServiceEx on OES Linux supports only NWSM_AUTH_RAW_DATA optionFlag in this developer release.

RELATED DOCUMENTATION

Following references point to the SMS API documentation that ships along with this developer release.

- For the following Connection related API refer the Connection Functions section in the Target Services Functions chapter of the API Documentation
  - NWSMTSConnectToTargetServiceEx
  - NWSMTSConnectToTargerService
- For examples on how to use these API refer to the demonstration engine documentation and code.

Connection Specific Configuration

Background

Target services provide the functionality to control certain internal features and are exposed through command line options or through iManager. A change to any of the parameters impact all backup/restore operations done using that target service. However, in various scenarios, there is a strong need to have the ability to control these configuration parameters on a per connection basis rather than for the entire TSA instance.

Configuration support in SMS

To provide backup applications the feasibility of defining these parameters for a particular connection (instead of the TSA instance itself), NWSMTSConfigureTargetSerivce API has been provided. This API exposes parameters that need to be different for each connection. All parameters set via this API takes precedence over the options that may have been set with the command line or iManager.

Recommendation for backup applications

- Backup applications must set the NWSM_USE_CACHING_MODE to false while doing incremental or differential backups based on application filters rather than using the target service specific filters. This action prevents TSA read-ahead caching which can actually degrade performance if the backup application discards several data sets that were read ahead.
- Backup applications that implement their own internal filters (those that do not use the TSAs internal filters) should also set the NWSM_USE_CACHING_MODE to false to prevent unnecessary read-aheads that would in turn degrade performance.
- The API also supports several other parameters that can be controlled per connection which can be used based on the design and need of the backup application and its features.

NOTE: The configuration changes made for backup specific options are connection specific and take effect from the next NWSMTSScanDataSetBegin or NWSMTSScanDataSetContinue call only.

Related documentation

Following references point to the SMS API documentation that ships along with this developer release.

- For the following backup related API refer the Backup Functions section in the Target Services Functions chapter of the API Documentation
    - NWSMTSScanDataSetBegin

- NWSMTSScanDataSetEnd

- NWSMTSConfigureTargetService definition can be found in the Target Services Functions Chapter of the API Documentation

Conclusion

The SMS API framework is available on OES Linux in addition to NetWare with the OES release. SMS has a single consistent interface to backup file systems on OES NetWare and OES Linux and other Novell applications. The API set has also been enhanced in the recent past to include newer functionality. This document provided an overview of the functionality, recommendations for their usage and limitations. This is expected to help backup application development in parallel with concurrent development of SMS and NSS on OES Linux.

Appendix A - SMS binaries on OES Linux

This appendix introduces the binary implementations of TSA and SMDR on OES Linux.

Filename	Developer Kit Location	Description
smdrd	/opt/novell/sms/bin	SMDR runs in user address space as a daemon. The daemon waits for connection requests and spawns a worker thread in response to each connection request it receives from other SMDRs in the network. The load TSA option of smsconfig requests SMDR to dynamically load TSA into its address space. *smdrd --help* will display all the smdrd command options.
smsconfig	/opt/novell/sms/bin	A loader has been implemented that will help register a particular TSA with SMDR. Each target that has a TSA will need to register is service name to enable routing of requests to it. Refer smsconfig man page for a detailed description of its command line options. smsconfig --help will display all the smsconfig command options. `smsconfig -l tsafs` This command registers the file system service as “Linux File System” and identifies libtsafs.so as the corresponding shared object that provides the service.
libsmdr.so	/opt/novell/lib	This is a shared library that exports the SMS APIs. The 32 bit applications working with SMS need to link with this library.
libsmdr.so	/opt/novell/lib64	This is a shared library that exports the SMS APIs. The 64 bit applications on a 64 bit machine working with SMS need to link with this library.

Filename	Developer Kit Location	Description
libtsafs.so	/opt/novell/lib	The TSA shared library implements all the necessary service functions to backup a target. Every target service will have to be loaded using smsconfig which will register the name of the service provided by the library. libtsafs.so implements the service functions to backup a file system.
libsmsut.so	/opt/novell/lib	This shared library contains the definitions of all the utility functions supported by SMS on 32 bit machine.
libsmsut.so	/opt/novell/lib64	This shared library contains the definitions of all the utility functions supported by SMS on 64 bit machine.