WS-Man File Transfer in 14th Generation Dell EMC PowerEdge Servers

This technical white paper describes the use of iDRAC9 WS-Man API client-local file streaming to simplify management automation.

Dell Engineering
June 2017

Authors

Sreelakshmi V

Hari Venkatachalam

Chinmay Hegde

Revisions

Date	Description
February 2017	Initial release
March 2017	Internal review 1
April 2017	Internal review 2
June 2017	Approved

Contents

1 Executive summary

Historically, administrators have relied upon network file shares such as CIFS and NFS when importing and exporting information from the integrated Dell Remote Access Controller (iDRAC) with Lifecycle Controller. Changing security considerations have caused many data centers to limit the use of such network shares, necessitating an API-based data transfer method for locally stored files.

This technical white paper describes enhancements to the iDRAC WS-Man API enabling the “streaming” of information to/from files local to the WS-Man client, removing the necessity of network file shares for a range of iDRAC operations.

2 Introduction

As the scale of deployment has grown for x86 servers, IT administrators have seen their scope expand from managing a handful of servers to hundreds or even thousands of servers. The deployment scale and the IT models have changed—from physical to virtual, from on-premises, to cloud, to hybrid cloud—leading to overall changes in the tools and processes of IT management.

In response to these changes, Dell EMC embeds standards-based management automation into every Dell EMC PowerEdge server—the iDRAC with Lifecycle Controller. Supporting automation of key management functions throughout the life of the server, iDRAC with Lifecycle Controller enables server deployment, monitoring, and update by using command line interface (CLI) and application programming interfaces (APIs) such as IPMI, WS-Man, and Redfish.

Many management automation functions require the import or export of information from iDRAC to a file. Historically, iDRAC supported these operations by using network file sharing methods such as CIFS and NFS but changing data center security considerations are leading administrators to limit the use of file shares.

To avoid the dependency on a network share, Dell EMC has enhanced the iDRAC WS-Man API for 14th generation PowerEdge servers to support a file transfer mechanism, enabling key management automation functions to be performed by “streaming” files local to and from the WS-Man client. This technical whitepaper provides information about the WS-Man file transfer mechanism and provides examples of use case scenarios that operate by using local files.

2.1 WS-Man file transfer prerequisites

- A software license (Base license) for 14th generation PowerEdge servers.
  For more information about managing licenses by using iDRAC GUI, click **Overview** → **Server Licenses** and see the iDRAC *Online Help*.
- The server must have a valid Service Tag (seven characters).
- User must have the login and configuration privileges. For export operations, only login privileges are required.
- iDRAC9 firmware version 3.00.00.00 or later must be installed.

2.2 Solution overview of WS-Man file transfer

The iDRAC WS-Man file transfer mechanism is based on File Transfer Protocol (RFC959). Based on the size, the file to be transferred is broken down and transmitted as a series of sizable chunks. At the receiving end the chunks are collated in the proper sequence to get the complete file.

For example, to transfer a 1KB file, management application can choose to break the file into 256 bytes and transfer it in four equal sized chunks.

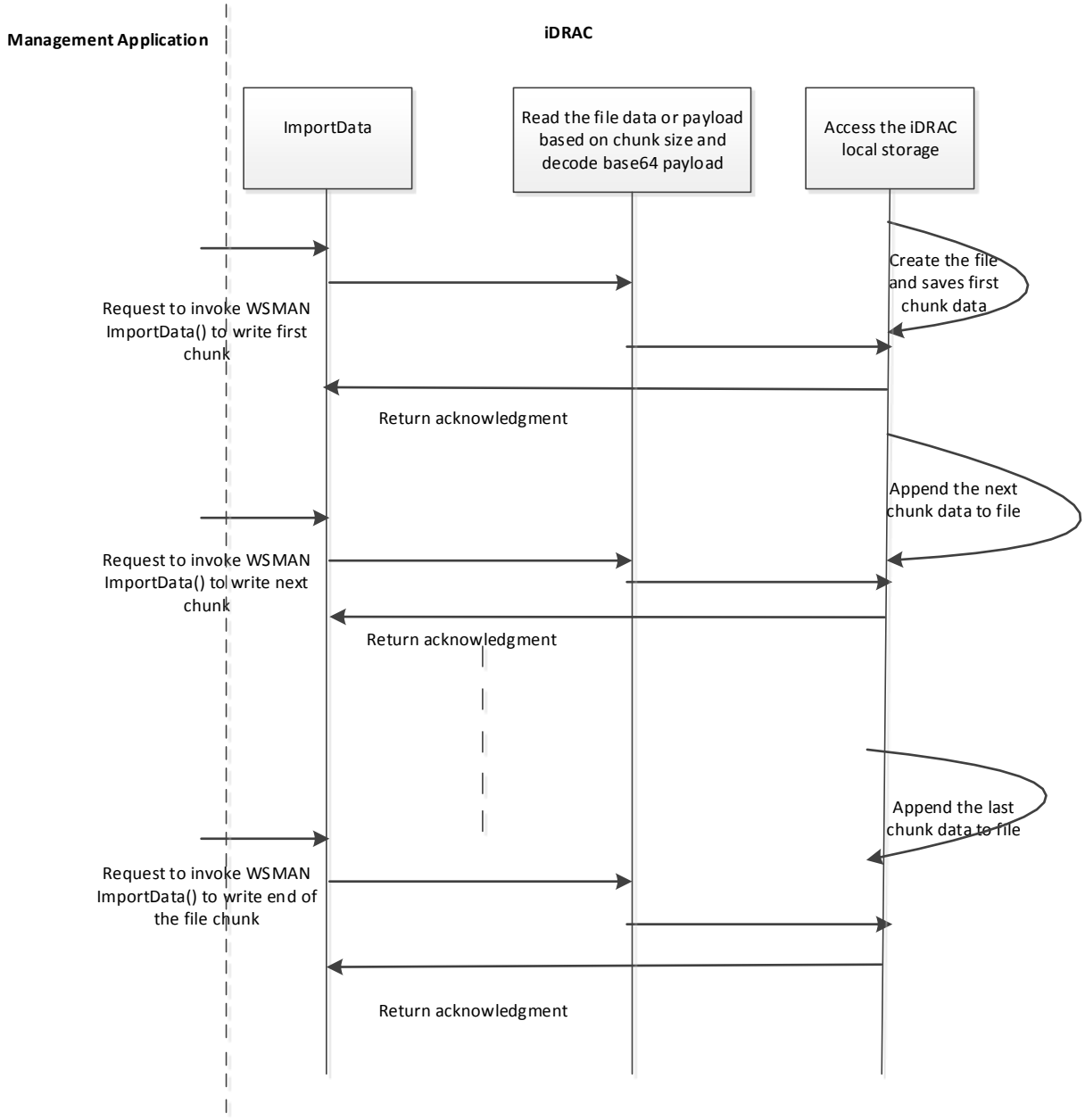
For iDRAC9 firmware version 3.00.00.00, the following management automation functions have been enhanced to support WS-Man file transfer in addition to network file share support:

Supported Files	**Purpose**
System Configuration Profile (SCP) file	Export and import of SCP files
Hardware Inventory	Export server hardware inventory from iDRAC
Lifecycle Controller Logs	Export Lifecycle Controller logs from iDRAC
Support Assist Collection	Export SupportAssist Collections from iDRAC
Diagnostics	Export Diagnostics results from iDRAC
Factory configuration	Export factory configuration files from iDRAC

2.2.1 ImportData mechanism

Import data refers to uploading files from a management application to a local folder in iDRAC. For example, if the management application wants to import a Server Configuration Profile (SCP), it can invoke the `ImportData` method defined in the `DCIM_iDRACCardService` class.

Sequence diagram for the “ImportData” method is as follows:



Management applications can transfer whole an entire file in a single message if the file size is small otherwise, the application requires to break down the file and transfer it in sequence of sizable packets. Based on an identifier for the sequence of packets, the iDRAC can reform the entire file at its end.

The sequence of the file transfer to iDRAC must be broken down into first packet, subsequent packets, and finally an iDRAC9 firmware version 3.00.00.00 or later must be installed. The format for each packet is described here.

First Packet

Represents the start of file transfer to iDRAC. The values for the method parameters when the first packet is transmitted is as follows.

- Management application request contains the following parameters to represents first packet

Parameters	Description
File Size	Size of the imported file
Session ID	Empty in the first packet
TxfrDescriptor	Marked transfer descriptor as “Start of transmit”
Chunk Size	Size of the chunk data or pay load
PayLoad	Management application will send the payload as Base64 encoded format

- Method response contains the following parameters to acknowledge on receiving first packet and continue for further transfers.

Session ID: A unique session ID is generated and it will be used in the subsequent packets. This ensures that blocks are pertaining to the same file. After receiving the response, iDRAC will copy the payload into the file.

Subsequent packet

Represents the normal transmission or subsequent data blocks to iDRAC. The values for the method parameters when the subsequent packet is transmitted is as follows.

- Management application request contains the following parameters to represents subsequent packets:

Parameters	Description
`Session ID`	Management application will specify the session ID which is retrieved in response from iDRAC and file name.
`ChunkSize`	Size of the chunk data or pay load.
`TxfrDescriptor`	Marked transfer descriptor as “Normal transmission”.
`PayLoad`	Management application will send the payload as `Base64` encoded format.

- The response will contain the same session ID.

End of file packet

It represents the last packet in the file transfer sequence. The iDRAC will stop receiving packets and closes the file. Hereafter, if a request with the same session ID and payload with subsequent or first packet format is sent to iDRAC, an appropriate error is returned.

Management application request contains the following parameters to represent end of packet:

Parameters	Description
`TxfrDescriptor`	Marked transfer descriptor as “End of packet”.
`CRC`	Management application will send the CRC for full file.
`PayLoad`	Management application will send the payload as Base64 encoded format.

The response will contain the same session ID.

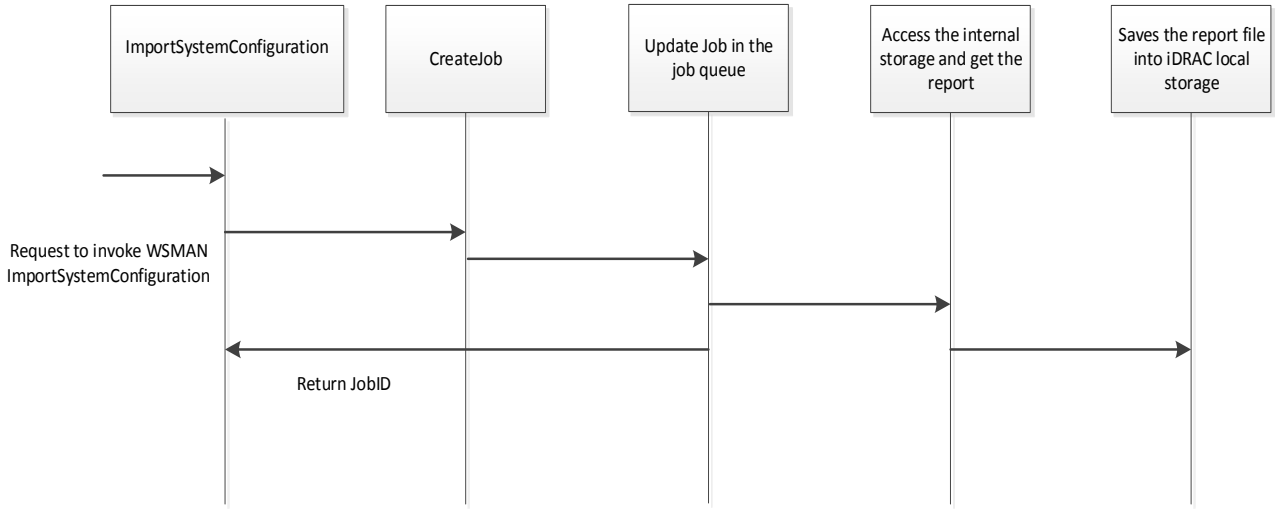
After completion of file transfer, management application invokes the existing method `ImportSystemConfiguration()` where share type is equal to local to perform system configuration profile updates.

Note: For more information about method input/output parameters, see the *Dell-iDRACCardProfile* document available on the Dell TechCenter.

The following methods have been enhanced for iDRAC9 to support files with share type of `local`:

Methods	Input Parameter (Unit16)
DCIM_LCService.ImportSystemConfiguration	ShareType = 4 for local
DCIM_LCService.ImportSystemConfigurationPreview	ShareType = 4 for local

Sequence diagram for `ImportSystemConfiguration()`:



Note: For more information about method input/output parameters, see the *Dell_LCManagementProfile* doc available on the Dell TechCenter.

2.3 Handling payload data during import operation

The WS-Man file transfer payload must be `Base64` encoded format because of inherent limitations of XML (SOAP) in handling binary data. To handle the `Base64` encoded data, the method requires a new line entry after each 64 characters in the Base64 payload and must provide chunk size as per the `Base64` standards.

Details about a new line entry after 64 characters:

- **The** management application needs to enter a new line after every 64 characters in the payload.

Note: It is not recommended to enter new line at the last line in payload.

For example:

```
<p:Payload>UmF3RXZlbnREYXRhPgogICAgPENvbW1lbnQvPgogICAgPFNvdXJjZS8+CiAgPC9F
dmVudD4KICA8RXZlbnQgQWdlbnRJRD0iV1NNQU4iIENhdGVnb3J5PSJBdWRpdCIg
U2V2ZXJpdHk9IkluZm9ybWF0aW9uYWwiIFRpbWVzdGFtcD0iMjAxNi0wNy0wN1Qw
NTowMDowMS0wNTAwIiBTZXF1ZW5jZT0iMTUiPgogICAgPE1lc3NhZ2U+VGhlIG9w
ICAgPFJhd0V2ZW50RGF0YS8+CiAgICA8Q29tbWVudC8+CiAgICA8U291cmNlLz4K
ICA8L0V2ZW50Pgo8L0xDTG9nRXZlbnRzPgo=</p:Payload>
```

- If the payload size is less than or equal to 64 characters, application must enter a new line.

  For example:

  ```
  <p:Payload>PD94bWwg

  </p:Payload>
  ```

Details about Input parameter chunk size:

- `ChunkSize` is calculated on the Base64 encoded payload.
- `ImportData` method decodes the Base64 payload and saves it in the iDRAC local folder.
- `Chunk size` (excluding new line entry) must be divisible by 4 to avoid extra padding while converting the Base64 to binary data.

  For example: binary size = (ChunkSize/4)*3

2.3.1 ExportData mechanism

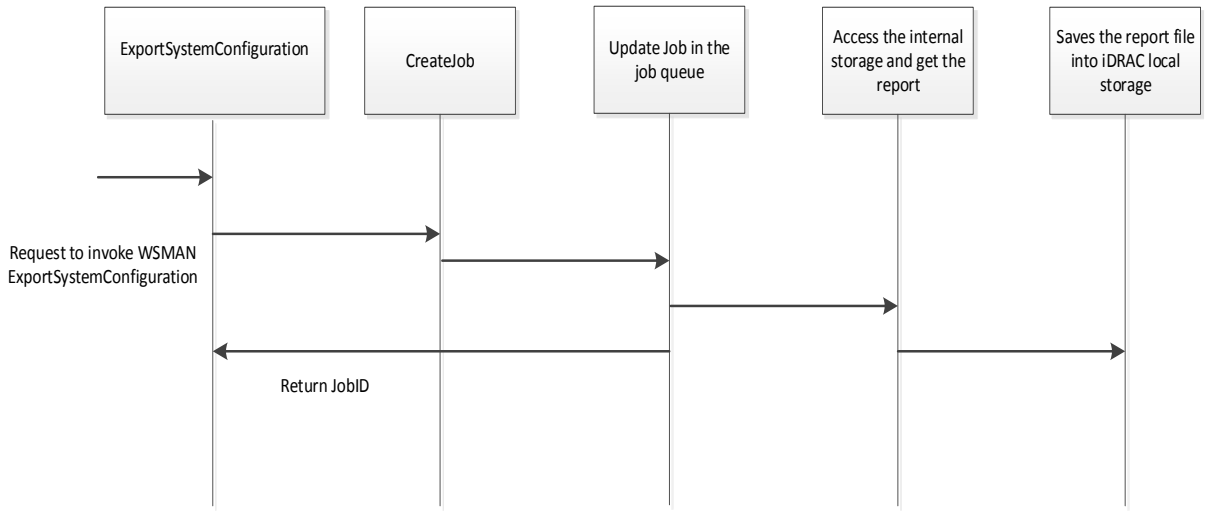
Export data refers to the download of files from iDRAC to a file folder local to the management application. For example, if the management application wants to export the Server Configuration Profile (SCP) file, it can invoke the `ExportData` method defined in the `DCIM_iDRACCardService` class. However, before invoking the `ExportData` method, the management application must invoke the existing export methods where share type equal to `local` in order to export file into the iDRAC local folder.

The following export methods have been enhanced for iDRAC9 to the local share type:

Method	**Input Parameter (Unit16)**
DCIM_LCService.ExportSystemConfiguration	ShareType = 4 for local
DCIM_LCService.ExportLCLog	ShareType = 4 for local
DCIM_LCService.ExportCompleteLCLog	ShareType = 4 for local
DCIM_LCService.ExportePSADiagnosticsResult	ShareType = 4 for local
DCIM_LCService.SupportAssistCollection	ShareType = 4 for local
DCIM_LCService.ExportHWInventory	ShareType = 4 for local
DCIM_LCService.ExportVideoLog	ShareType = 4 for local
DCIM_LCService.ExportFactoryConfiguration	ShareType = 4 for local

Note: For more information about method input/output parameters, see the *Dell_LCManagementProfile* document available at the Dell TechCenter.

The sequence diagram for existing export methods is as follows:
Example for `ExportSystemConfiguration`



After the completion of the `ExportSystemConfiguration` method, the management application must invoke the `ExportData` method to save SCP file into their local folder.

As mentioned in the `ImportData` method, management applications can get entire file in single transfer, if the file size is small. In this case, the iDRAC breaks down the large file and transfers it in a sequence of blocks. Based on an identifier for the sequence of blocks, the management application can reform the entire file at its end.

The sequence of the file transfer from iDRAC has to be broken down into first packet, subsequent packets, and end of file packet. The format for each packet is described here:

First Packet

It represents the start of the file transfer from iDRAC. The values for the method parameters when the first packet is transmitted is as follows:

a. Management application Request contains the following parameters to get first packet.

Parameters	Description
InChunkSize	The size of each block the management application expects to receive
SessionID	Empty in the first packet
FileOffset	Offset of the file. "FileOffset" will be "0" for the first chunk.
TxDataSize	Transmitted data size. "TxDataSize" will be "0" for the first chunk.

b. Method response contains the following parameters to represent start of transmission and continue for further transfers.

Parameters	Description
FileSize	Size of the exported file
SessionID	A unique session ID is generated and it will be used in the sub-sequent packets. This ensures that blocks are pertaining to the same file.
TxfrDescriptor	Marked as "Start of transmit"
ChunkSize	Size of chunk data or pay load.
CRC	CRC for full file
FileOffset	Current position (offset) of the exported file
TxDataSize	Transmitted data size of the export file
PayLoad	iDRAC will send the payload as Base64 encoded format

Subsequent packet

It represents the normal transmission or sub sequent data blocks from the iDRAC. The values for the method parameters when the subsequent packet transmitted is as follows.

Management application request contains the following parameters to get subsequent packets:

Parameters	Description
SessionID	Management application will specify the session ID which is received in response from iDRAC.
InChunkSize	The chunk size for the next block to be received.
FileOffset	File offset for the next block to be received.
TxDataSize	Management application needs to specify the received transmitted data size of the exported file.

Method response contains the following parameters to represent start of transmission and continue for further transfers:

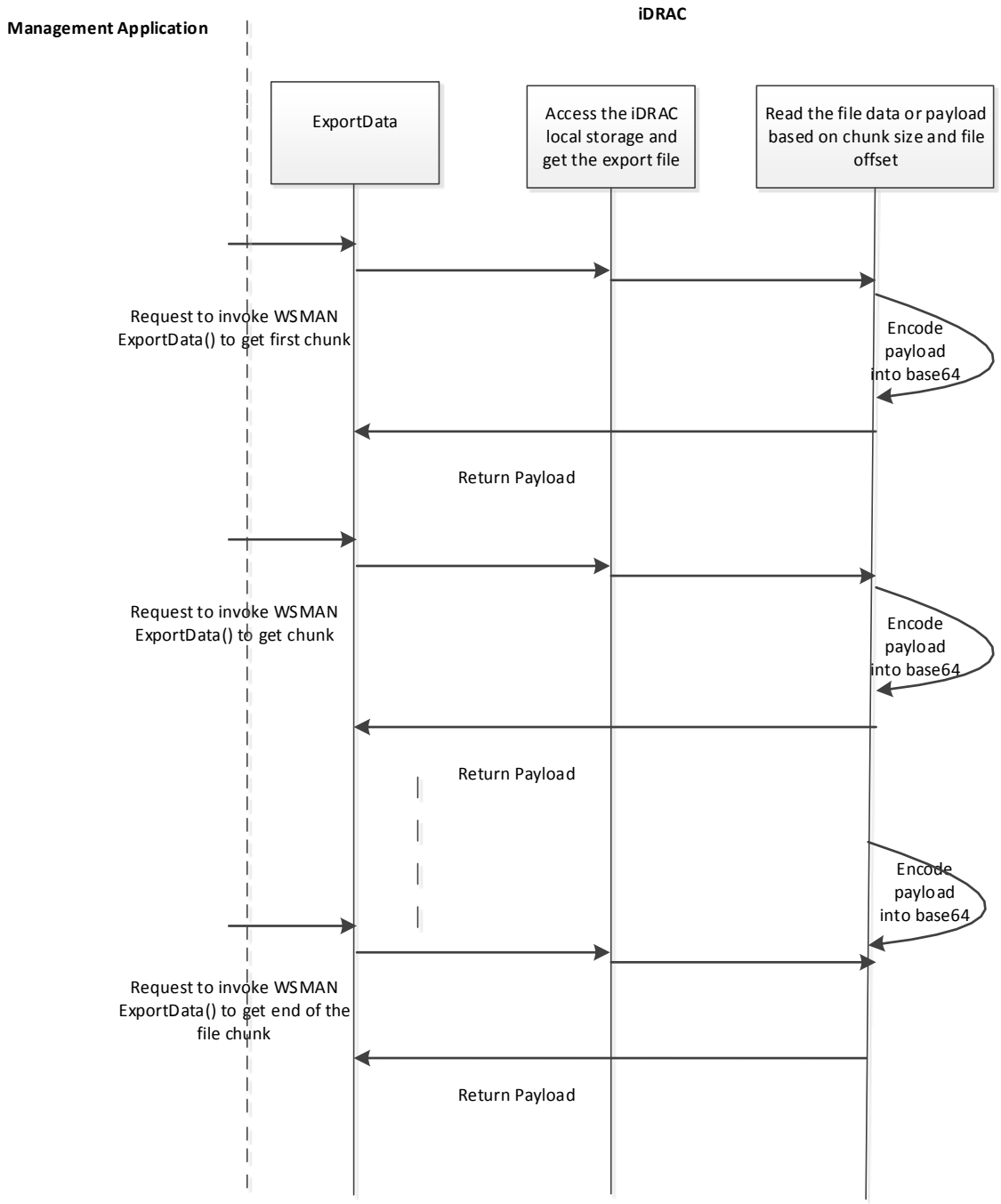
Parameters	Description
FileSize	Size of the exported file
SessionID	Method response contain the same session ID.
TxfrDescriptor	Marked transfer descriptor as “Normal transmission”
ChunkSize	Size of chunk data or pay load.
FileOffset	Current position (offset) of the exported file
TxDataSize	Transmitted data size of the export file
PayLoad	iDRAC will send the payload as Base64 encoded format

End of file packet

It represents the last packet in the file transfer sequence. When the end of transmission is reached, the iDRAC will mark the transfer descriptor as “End of packet”. The Management application after receiving this will stop requesting for further packets.

Note: For more information about method input/output parameters, see the *Dell- iDRACCardProfile* document at Dell TechCenter.

The sequence diagram for "`ExportData`" method is described here:



2.4 Handling of payload data during export operation

The management application will get the Base64 payload from iDRACby invoking the `ExportData` method. The payload contains the new line entry after every 64 characters.

- Details on export chunk size:

  The `ExportData` method has Input and output parameters for chunk size.

  - `InChunkSize` is the input parameter in `ExportData` method. It represents size of binary payload. Management application will divide the binary file size into number of chunks and pass the chunk size value to iDRAC through in input parameter `InChunkSize`.
  - `InChunkSize` must be divisible by 3 to avoid extra padding while converting the binary data to Base64. For example, Base64 size = 4* (InChunkSize/3)
- `ChunkSize` is the output parameter in ExportData method. It represents the size of the exported Base64 payload including new lines.

2.4.1 ClearTransferSession

The purpose of the `ClearTransferSession` method is error handling. As per the design, any import or export file will be available in the iDRAC local folder for a duration of 30 minutes (timeout), and till the completion of 30 min, concurrent requests are not allowed for the same file type. In case of issues during file transfer—such as partial file transfer or erroneous file—will be stored in the iDRAC local folder and further file transfers will be blocked till timeout (30 min) is elapsed.

To avoid this situation, WS-Man provides the `ClearTransferSession` method to restart file transfers and clear operation performed independent of session ID.

The `ClearTransferSession` method will delete the files based on the input parameters file type and file operation.

Note: For more information about method input/output parameters, see the *Dell-iDRACCardProfile* document available at the Dell TechCenter.

2.5 Constrains for WS-Man transfer

- No file transfer data persists if the iDRAC is reset.
- For the `ImportData` method, if some packets are lost in between, the management application must restart the transfer.
- If a legacy export method jobs fail when the share type option `local` because of any issues, the management application must invoke `ClearTransferSession` to clear the session before starting an export method with the share type to `local`.
- After completion of the `ImportData` method, the management application must invoke the `ImportSystemConfiguration` method to import the file within 30 minutes. Else, file may be overwritten or deleted by another `ImportData` request with same file type.
- Similarly for export, the management application must invoke the `ExportData` method to get the file to their local folder within 30 minutes after completion of export method with the share type to `local`.
- To transfer a binary payload as part of the WS-Man SOAP XML, the binary data must be converted to a Base-64 format.
- Multiple transfers on same file type is not allowed at the same.

2.6 WS-Man file transfer PowerShell

The following example illustrates WS-Man file transfer requests using PowerShell scripting. First, a session connection is established:

```
$SessionOtions=New-CimSessionOption -SkipCACheck -SkipCNCheck -SkipRevocationCheck -
Encoding Utf8 -UseSsl


$Authentication_Credentials=New-CimSession -Authentication Basic -Credential <%UserName%>
-ComputerName <%IPAddress%> -Port 443 -SessionOption $SessionOtions

 $Inst=Get-CimInstance -CimSession $Authentication_Credentials -ResourceUri
"http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardService" -
Namespace root/dcim -Filter 'SystemCreationClassName="DCIM_ComputerSystem" AND
SystemName="DCIM:ComputerSystem" AND CreationClassName="DCIM_iDRACCardService" AND
Name="DCIM:iDRACCardService"'
```

2.6.1 ImportData

To import files to the iDRAC, invoke the `ImportData` method. Sample scripts are given here.

---

1. Transfer first packet. Ensure that:
    - The InSessionID=empty
    - CRC=Use MD5 algorithm to calculate the CRC of the entire file.
    - TxfrDescriptor=1 (for the first packet)
    - PayLoadEncoding=Currently, only Base64 is supported which is indicated by '2'.

```
$input_parameters=@{InSessionID="";ChunkSize="<chunk size in
bytes>";TxfrDescriptor="1";FileSize="<file size in bytes>";FileType="<type of file to be
imported>";CRC="<crc of the entire file>";PayLoaEncoding="2";Payload="<Base64 payload>"}

Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ImportData -Arguments $input_parameters | Format-Custom

class CimMethodResult#DCIM_iDRACCardService#ImportData
{
  Message = The Import File operation is successfully completed.
  MessageID = RAC077
  ReturnValue = 0
  SessionID = 79648039
}
```

---

2. Transfer intermediate packet. Ensure that:
    - The InSessionID=Enter the ID that you receive as a response from the first packet. For example, here it is `79648039`.
    - TxfrDescriptor=2 ('2' is a constant for intermediate packets)

```
$input_parameters=@{InSessionID="<Session ID upon receiving first packet
>";ChunkSize="<chunk size in bytes>";TxfrDescriptor="2";FileSize="<file size in
bytes>";FileType="="<type of file to be imported>";PayLoadEncoding="2";Payload="<Base64
payload>"}

Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ImportData -Arguments $input_parameters | Format-Custom


class CimMethodResult#DCIM_iDRACCardService#ImportData
{
  Message = The Import File operation is successfully completed.
  MessageID = RAC077
  ReturnValue = 0
  SessionID = 79648039
}
```

3. Transfer end of file packet. Ensure that:
    - The InSessionID=Enter the ID that you receive as a response from the first packet. For example, here it is `79648039`.
    - TxfrDescriptor=3

```
---------------------------------------------------------------------------------
$input_parameters=@{InSessionID="<Session ID upon receiving first packet
>";ChunkSize="<chunk size in bytes>";TxfrDescriptor="3";FileSize="file size in
bytes";FileType="<type of the file to be imported>"; PayLoadEncoding="2";Payload=”<Base64
payload>”}

Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ImportData -Arguments $input_parameters | Format-Custom

class CimMethodResult#DCIM_iDRACCardService#ImportData
{
  Message = The Import File operation is successfully completed.
  MessageID = RAC077
  ReturnValue = 0
  SessionID = 79648039
}
```

2.6.1.1 ImportSystemConfiguration

After successfully importing the file, invoke the `ImportSystemConfiguration` method by using the sample script given here.

```
$Inst=Get-CimInstance -CimSession $Authentication_Credentials -ResourceUri
"http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService" -Namespace
root/dcim -Filter 'SystemCreationClassName="DCIM_ComputerSystem" AND
CreationClassName="DCIM_LCService" AND SystemName="DCIM:ComputerSystem" AND
Name="DCIM:LCService"'

$input_parameters=@{ShareType="4"}


Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ImportSystemConfiguration -Arguments $input_parameters | Format-Custom

class CimMethodResult#DCIM_LCService#ImportSystemConfiguration
{
  Job =
    class CimInstance#root/dcim/Job
    {
      EndpointReference =
        class CimInstance#root/dcim/DCIM_LifecycleJob
        {
          InstanceID = JID_944134734141
        }
    }
  ReturnValue = 4096
}
```

Checking Job Status:

```
-------------------------------------------------
Get-CimInstance -CimSession $Authentication_Credentials -ResourceUri
"http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/DCIM_LifeCycleJob" -Namespace
"root/dcim" -Query "SELECT * FROM DCIM_LifeCycleJob Where InstanceID='JID_944134734141'"

ElapsedTimeSinceCompletion : 0
InstanceID                 : JID_944134734141
JobStartTime               : NA
JobStatus                  : Completed
JobUntilTime               : NA
Message                    : Successfully imported and applied Server Configuration
Profile.
MessageArguments           : NA
MessageID                  : SYS053
Name                       : Import Configuration
PercentComplete            : 100
```

2.6.2 ExportData

To export files from the iDRAC, invoke the `ExportData` method. Sample scripts are given here.

```
$Inst=Get-CimInstance -CimSession $Authentication_Credentials -ResourceUri
"http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardService" -
Namespace root/dcim -Filter 'SystemCreationClassName="DCIM_ComputerSystem" AND
SystemName="DCIM:ComputerSystem" AND CreationClassName="DCIM_iDRACCardService" AND
Name="DCIM:iDRACCardService"'
-------------------------------------------------
```

1. Transfer first packet. Ensure that:
   - The InSessionID=empty
   - TxDataSize = Zero (0) for the first packet.
   - FileOffset =Zero (0) for the first packet

```
-------------------------------------------------
$input_parameters=@{FileType=”<type of the file to be
exported>”;TxDataSize=”0”;FileOffset=”0”;InChunkSize="<chunk size in
bytes>";InSessionID=""}

Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ExportData -Arguments $input_parameters | Format-Custom


class CimMethodResult#DCIM_iDRACCardService#ExportData
{
  CRC = cd9495089cd94e739a6d8ee24e5b2a49 //MD5 checksum of the exported file
  ChunkSize = 80000
  FileSize = 141517
  Message = The Export File operation is successfully completed.
  MessageID = RAC080
  PayLoad = "<%Base64 payload %>"
  PayLoadEncoding = 2
  RetFileOffset = 60000
```

```
  RetTxDataSize = 80000
  ReturnValue = 0
  SessionID = 3365579175      //Unique session ID to maintain further transfers on the
                              same session.
  TxfrDescriptor = 1    //"1" indicates the start of the file transfer.
}
```

---

2. Transfer intermediate packet. Ensure that:
    - The InSessionID= Enter the ID that you receive as a response from the first packet. For example, here it is `3365579175`.
    - TxDataSize = Enter the value (`RetTxDataSize`) that you received as a response from the earlier packet. Here, it is 80,000.
    - FileOffset = Enter the value (`RetFileOffset`) that you received as a response from the earlier packet. Here, it is 60,000.

```
$input_parameters=@{FileType='1';TxDataSize='80000';FileOffset='60000';InChunkSize="60000
";InSessionID="3365579175"}

Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ExportData -Arguments $input_parameters | Format-Custom
class CimMethodResult#DCIM_iDRACCardService#ExportData
{
  ChunkSize = 80000
  FileSize = 141517
  Message = The Export File operation is successfully completed.
  MessageID = RAC080
  PayLoad = "<% Base64 payload %>"
  PayLoadEncoding = 2
  RetFileOffset = 120000
  RetTxDataSize = 160000
  ReturnValue = 0
  SessionID = 3365579175
  TxfrDescriptor = 2          //'2' indicates the intermediate file operations
}
```

---

3. Transfer end of the packet.

```
$input_parameters=@{FileType='1';TxDataSize='160000';FileOffset='120000';InChunkSize="600
00";InSessionID="3365579175"}

Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ExportData -Arguments $input_parameters | Format-Custom
class CimMethodResult#DCIM_iDRACCardService#ExportData
{
  ChunkSize = 28692
  FileSize = 141517
```

```
  Message = The Export File operation is successfully completed.
  MessageID = RAC080
  PayLoad = "<% Base64 payload %>"
  PayLoadEncoding = 2
  RetFileOffset = 141517
  RetTxDataSize = 188692
  ReturnValue = 0
  SessionID = 3365579175
  TxfrDescriptor = 3    //'3' indicates the end of file transfer operation
}
```

2.6.2.1 ExportSystemConfiguration

Before exporting the file from iDRAC, invoke the `ExportSystemConfiguration` method by using the sample script given here.

```
$Inst=Get-CimInstance -CimSession $Authentication_Credentials -ResourceUri
"http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService" -Namespace
root/dcim -Filter 'SystemCreationClassName="DCIM_ComputerSystem" AND
CreationClassName="DCIM_LCService" AND SystemName="DCIM:ComputerSystem" AND
Name="DCIM:LCService"'


$input_parameters=@{ShareType="4"}

Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ExportSystemConfiguration -Arguments $input_parameters | Format-Custom

class CimMethodResult#DCIM_LCService#ExportSystemConfiguration
{
  Job =
    class CimInstance#root/dcim/Job
    {
      EndpointReference =
        class CimInstance#root/dcim/DCIM_LifeCycleJob
        {
          InstanceID = JID_944130267152
        }
    }
  ReturnValue = 4096
}
```

Checking Job Status:

```
Get-CimInstance -CimSession $Authentication_Credentials -ResourceUri
"http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/DCIM_LifeCycleJob" -Namespace
"root/dcim" -Query "SELECT * FROM DCIM_LifeCycleJob Where InstanceID='JID_944130267152'"
ElapsedTimeSinceCompletion : 0
InstanceID                 : JID_944130267152
JobStartTime               : NA
JobStatus                  : Completed
JobUntilTime               : NA
Message                    : Successfully exported Server Configuration Profile
MessageArguments           : NA
MessageID                  : SYS043
```

```
Name                          : Export Configuration
PercentComplete               : 100
```

2.6.3 ClearTransferSession

To delete the import or export files stored in the iDRAC local folder, invoke the `ClearTransferSession` method by using the sample script given here.

```
$Inst=Get-CimInstance -CimSession $Authentication_Credentials -ResourceUri
"http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardService" -
Namespace root/dcim -Filter 'SystemCreationClassName="DCIM_ComputerSystem" AND
SystemName="DCIM:ComputerSystem" AND CreationClassName="DCIM_iDRACCardService" AND
Name="DCIM:iDRACCardService"'

$input_parameters=@{FileOperation="1";FileType='1'}

Invoke-CimMethod -InputObject $Inst -CimSession $Authentication_Credentials -MethodName
ClearTransferSession -Arguments $input_parameters | Format-Custom

class CimMethodResult#DCIM_iDRACCardService#ClearTransferSession

{

  Message = The Clear Transfer Session operation is successfully completed.

  MessageID = RAC084

  ReturnValue = 0

}
```

3 Conclusion

The WS-Man file transfer feature enables the management application to remotely initiate a file transfer to and from iDRAC9 without requiring network shares such as CIFS or NFS. WS-Man file transfer is more secure when compared to network share because file transfers happen through HTTPs.

Management application can transfer the file from iDRAC to their local share or vice versa by using simple WS-Man commands or scripts.

4 References

- WS-Man Interface Guide for Linux: http://en.community.dell.com/techcenter/extras/m/white_papers/20066176.aspx
- WS-Man Interface Guide for Windows: http://en.community.dell.com/techcenter/extras/m/white_papers/20066174.aspx
- WS-Man command line open source for Linux (OpenWSMan): http://sourceforge.net/projects/openwsman/
- OpenWSMan installation instructions: http://en.community.dell.com/techcenter/systems-management/w/wiki/3567.instructions-installing-openwsman-cli-on-linux.aspx
- WS-Man command line for Windows (winrm): http://msdn.microsoft.com/en-us/library/windows/desktop/aa384291(v=VS.85).aspx
- WS-Man scripts for the Dell Lifecycle Controller: http://en.community.dell.com/techcenter/systems-management/w/wiki/scripting-the-dell-lifecycle-controller.aspx