NetBrain Enterprise Edition 6.1 System Integration Guide

1. SOAP Web Service Based on API for Device Data Import in EE6.1
   1.1 Background: The Need for Data Import
   1.2 Data Import Web Service API in EE6.1
   Login
   AddDeviceByConfig
   AddIPforDiscovery
   RemoveAllIPforDiscovery
   RemoveIPforDiscovery
   CreateCustomizedDeviceAttribute
   SetCustomizedDeviceAttribute
   GetCustomizedDeviceAttribute
   SetDeviceAttribute
   GetDeviceAttribute
   SetNetWorkServerOfDevice
   DeleteDeviceByHostName
   RebuildL3Topology
   SetDeviceSitePath
   RebuildSiteTopology
   Logout

2. Remote Query of PostgreSQL Database
   2.1 Introduction
   2.2 Configure PostgreSQL
   2.3 Add the read only User to the database
   2.4 Frequently used tables
   2.5 SQL Query Examples
   2.6 Device Type

3. File-based inventory data synchronization (HPNA integration)
   3.1 Introduction
   3.2 Define diagnostics to retrieve the data in HP NA
   3.3 Define tasks to export live data from HP NA to a folder
3.4 Schedule the schedule discovery task to import the data from the folder............................................41

4. Expose Dynamic Mapping to Third Party Tools (Netcool Integration) ..........................................................43
   4.1 Introduction ..................................................................................................................................................43
   4.2 Setup Netcool Integration ..........................................................................................................................43

5. Drive Third Party Tools from the Map.........................................................................................................48

6. Integrate with inventory management tools ..................................................................................................48

Appendix: Asset Report Storage ......................................................................................................................49
   1. Device Report ................................................................................................................................................49
   2. Interface Report ..........................................................................................................................................53
   3. Module Report .............................................................................................................................................57
   4. VSS Report ..................................................................................................................................................58
   5. Stackable Switch Report ...............................................................................................................................58
   6. Site Report ....................................................................................................................................................58
1. SOAP Web Service Based on API for Device Data Import in EEv6.1

1.1 Background: The Need for Data Import

Large numbers of enterprise customers prefer to have centralized database maintain basic device information, (eg. IPAM or CMDB) as a single source of truth on their network assets. They prefer to have the rest of the NMS systems use the data from this central system without conducting independent network discovery on their own in order to improve consistency across the company and reduce overhead to their network.

Periodically import and update on configuration file, IP addresses, device attributes, site structure, show command outputs, etc.

1.2 Data Import Web Service API in EE6.1

WSDL interface: http://(your server ip)/workspace/(workspace ID)/nbws3rdparty.asmx?WSDL, for instance, http://10.10.10.197/workspace/1/nbws3rdparty.asmx?WSDL (WSDL is disabled by default in EE6.0a and later versions for security, to enable it, please use the solution: navigate to \NetBrain\Enterprise Server\Workspaces\Workspace1\WebServer\Web.config, double-click Web.config to open it and delete the field <remove name="Documentation" / > in the web config window.

- 15 data import APIs supported as the initial phase of NetBrain API support. More APIs will be provided in the future release.
  - Login
  - AddDeviceByConfig
  - AddIPforDiscovery
  - RemoveAllIPforDiscovery
  - RemoveIPforDiscovery
  - CreateCustomizedDeviceAttribute
  - SetCustomzedDeviceAttibute
  - GetCustomDeviceAttibute
  - SetDeviceAttribute
  - GetDeviceAttribute
  - SetNetWorkServerOfDevice
All users have the right to execute the APIs Login, GetCustomDeviceAttribute, GetDeviceAttribute and Logout. But only the users with role permissions System Management or Shared Workspace Management have the right to execute the others.

For each of SOAP API values, we will provided examples and descriptions, please refer to http://(your server_ip)/workspace/(workspace ID)/nbws3rdparty.asmx? for details.

For detailed SOAP API format, please refer to the API WSDL.

Mode:
- Blocked: it indicates that the API sends data to NetBrain server and waits for responding.
- Not blocked: it indicates that the API sends data to NetBrain server and does not wait for responding.

Login

Log into NetBrain SOAP API. To enhance the security of network communication, we recommend to choose HTTPS to have the communication encrypted.

Mode: Blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```xml
POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Login"

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <Login xmlns="http://tempuri.org/">
      <username>string</username>
      <password>string</password>
    </Login>
  </soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <LoginResponse xmlns="http://tempuri.org/">
      <LoginResult>
        <Status>Success</Status> or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
        <Message>string</Message>
      </LoginResult>
      <identityKey>string</identityKey>
    </LoginResponse>
  </soap:Body>
</soap:Envelope>

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;username&gt;netbrain&lt;/username&gt;</td>
<td>Input the username to log to the SOAP API. Type: string.</td>
</tr>
<tr>
<td>&lt;password&gt;netbrain&lt;/password&gt;</td>
<td>Input the password to log to the SOAP API. Type: string.</td>
</tr>
<tr>
<td>Output</td>
<td>Description</td>
</tr>
<tr>
<td>&lt;identityKey&gt;29efca900e186f3dd&lt;/identityKey&gt;</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
</tbody>
</table>

**Note: The 'identitykey' will expire in 60 minutes, please re-login to generate a new one if needed.**

**AddDeviceByConfig**

Add device into NetBrain by importing configuration files. Assume NetBrain will still be able to directly access the device via the credentials provided for data retrieval.

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```xml
POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/AddDeviceByConfig"

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <AddDeviceByConfig xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
    </AddDeviceByConfig>
  </soap:Body>
</soap:Envelope>
```
<configItems>
   <DeviceConfigItem>
      <bytesFromConfigFile>base64Binary</bytesFromConfigFile>
      <strDeviceName>string</strDeviceName>
      <strUserName>string</strUserName>
      <strPassword>string</strPassword>
      <strEnableUserName>string</strEnableUserName>
      <strEnablePassword>string</strEnablePassword>
      <strSNMPRo>string</strSNMPRo>
      <strJumpbox>string</strJumpbox>
      <strNetworkServer>string</strNetworkServer>
   </DeviceConfigItem>
   <DeviceConfigItem>
      <bytesFromConfigFile>base64Binary</bytesFromConfigFile>
      <strDeviceName>string</strDeviceName>
      <strUserName>string</strUserName>
      <strPassword>string</strPassword>
      <strEnableUserName>string</strEnableUserName>
      <strEnablePassword>string</strEnablePassword>
      <strSNMPRo>string</strSNMPRo>
      <strJumpbox>string</strJumpbox>
      <strNetworkServer>string</strNetworkServer>
   </DeviceConfigItem>
   <DeviceConfigItem>
      <bytesFromConfigFile>base64Binary</bytesFromConfigFile>
      <strDeviceName>string</strDeviceName>
      <strUserName>string</strUserName>
      <strPassword>string</strPassword>
      <strEnableUserName>string</strEnableUserName>
      <strEnablePassword>string</strEnablePassword>
      <strSNMPRo>string</strSNMPRo>
      <strJumpbox>string</strJumpbox>
      <strNetworkServer>string</strNetworkServer>
   </DeviceConfigItem>
</configItems>

<isRebuild>boolean</isRebuild>
</AddDeviceByConfig>
</soap:Body>
</soap:Envelope>
<table>
<thead>
<tr>
<th>DeviceConfigItem(sample)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;identityKey&gt; 29efca900e186f3dd&lt;/identityKey&gt;</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
<tr>
<td>&lt;bytesFromConfigFile&gt;base64Binary&lt;/bytesFromConfigFile&gt;</td>
<td>UTF-8 and ANSI encodings are supported.</td>
</tr>
<tr>
<td>&lt;strDeviceName&gt; GW2Lab &lt;/strDeviceName&gt;</td>
<td>Input the hostname of the device which is expected to add into workspace. Type: string.</td>
</tr>
<tr>
<td>&lt;strUserName&gt; nb &lt;/strUserName&gt;</td>
<td>Input the username to log to the device. Type: string.</td>
</tr>
<tr>
<td>&lt;strPassword&gt; netbrain &lt;/strPassword&gt;</td>
<td>Input the password to log to the device. Type: string.</td>
</tr>
<tr>
<td>&lt;strEnableUserName&gt; nb &lt;/strEnableUserName&gt;</td>
<td>Input username to enter the privilege mode of the device. Type: string.</td>
</tr>
<tr>
<td>&lt;strEnablePassword&gt; netbrain1 &lt;/strEnablePassword&gt;</td>
<td>Input the password to enter the privileged mode of the device. Type: string.</td>
</tr>
<tr>
<td>&lt;strSNMPRo&gt; NB &lt;/strSNMPRo&gt;</td>
<td>Input the SNMP RO alias predefined in the Shared Network Settings. SNMPv3 is not supported in current release. Type: string.</td>
</tr>
<tr>
<td>&lt;strNetworkServer&gt; NB-NS &lt;/strNetworkServer&gt;</td>
<td>Input the Network server alias predefined in the Shared Network Settings. Type: string.</td>
</tr>
<tr>
<td>&lt;isRebuild&gt; true &lt;/isRebuild&gt;</td>
<td>true: Execute the L3 network topology rebuilding after importing the device. false: Do not execute the L3 network topology rebuilding after importing the device. Type: boolean.</td>
</tr>
</tbody>
</table>

Note:

1. The user needs to execute API RebuildL3Topology to refresh the new devices to workstation.
2. 10 devices are allowed to add at a time, please consult NetBrain Support for adjustment if necessary.

AddIPforDiscovery

Add IP to NetBrain database for future data collection. The IP will be called when doing Schedule Discovery tasks from NetBrain Server Workspace webpage.

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /workspace/1/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.123
Content-Type: text/xml; charset=utf-8
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AddIPforDiscovery xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <deviceIPItem>
        <strMgmtIP>string</strMgmtIP>
        <strDeviceName>string</strDeviceName>
        <strUserName>string</strUserName>
        <strPassword>string</strPassword>
        <strEnableUserName>string</strEnableUserName>
        <strEnablePassword>string</strEnablePassword>
        <strSNMPRo>string</strSNMPRo>
        <strNetworkServerName>string</strNetworkServerName>
        <strNetworkServerIP>string</strNetworkServerIP>
      </deviceIPItem>
      <deviceIPItem>
        <strMgmtIP>string</strMgmtIP>
        <strDeviceName>string</strDeviceName>
        <strUserName>string</strUserName>
        <strPassword>string</strPassword>
        <strEnableUserName>string</strEnableUserName>
        <strEnablePassword>string</strEnablePassword>
        <strSNMPRo>string</strSNMPRo>
        <strNetworkServerName>string</strNetworkServerName>
        <strNetworkServerIP>string</strNetworkServerIP>
      </deviceIPItem>
    </AddIPforDiscovery>
  </soap:Body>
</soap:Envelope>
<AddIPforDiscoveryResponse xmlns="http://tempuri.org/">
  <AddIPforDiscoveryResult>
    <Status>Success or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
    <Message>string</Message>
  </AddIPforDiscoveryResult>
</AddIPforDiscoveryResponse>
</soap:Body>
</soap:Envelope>

<table>
<thead>
<tr>
<th>DeviceIPItem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;identityKey&gt;29efca900e186f3dd&lt;/identityKey&gt;</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
<tr>
<td>&lt;strMgmtIP&gt;10.10.10.16&lt;/strMgmtIP&gt;</td>
<td>Input the management IP of the device. It will be added to database and called by Schedule Discovery task in Server Workspace. Type: string.</td>
</tr>
<tr>
<td>&lt;strDeviceName&gt;GW2Lab&lt;/strDeviceName&gt;</td>
<td>Input the hostname of the device which is expected to add into workspace. Type: string.</td>
</tr>
<tr>
<td>&lt;strUserName&gt;nb&lt;/strUserName&gt;</td>
<td>Input the username to log to the device. Type: string.</td>
</tr>
<tr>
<td>&lt;strPassword&gt;netbrain1&lt;/strPassword&gt;</td>
<td>Input the password to log to the device. Type: string.</td>
</tr>
<tr>
<td>&lt;strEnableUserName&gt;nb&lt;/strEnableUserName&gt;</td>
<td>Input the username to enter the privilege mode of the device. Type: string.</td>
</tr>
<tr>
<td>&lt;strEnablePassword&gt;netbrain1&lt;/strEnablePassword&gt;</td>
<td>Input the password to enter the privileged mode of the device. Type: string.</td>
</tr>
<tr>
<td>&lt;strSNMPRo&gt;NB&lt;/strSNMPRo&gt;</td>
<td>Input the SNMP RO alias predefined in the Shared Network Settings. SNMPv3 is not supported in current release. Type: string.</td>
</tr>
<tr>
<td>&lt;strNetworkServerName&gt;NB&lt;/strNetworkServerName&gt;</td>
<td>Input the Network Server alias predefined in the Shared Network Settings. Type: string</td>
</tr>
<tr>
<td>&lt;strNetworkServerIP&gt;10.10.10.88&lt;/strNetworkServerIP&gt;</td>
<td>Input the IP address or Domain name of the Network Server predefined in the Shared Network Settings. Type: string</td>
</tr>
</tbody>
</table>

Note:
1. The strMgmtIP, strNetworkServerName and strNetworkServerIP are required, NetBrain will poll the settings from Shared Network Settings if the others are not correct or not set.
2. The API AddIPforDiscovery adds the DeviceIPItem to the table discovery_device_by_ip in Postgresql database. And the DeviceIPItem is only called by Schedule Discovery tasks from NetBrain Server Workspace webpage.
3. The items in table discovery_device_by_ip cannot be modified once they are added, the user needs to clean them firstly via API RemoveAllIPforDiscovery or RemoveIPforDiscovery and then add them again.
Remove all IP Items in the table discovery_device_by_ip in Postgresql database which are added by API AddIPforDiscovery.

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /workspace/1/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/RemoveAllIPforDiscovery"

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <RemoveAllIPforDiscovery xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
    </RemoveAllIPforDiscovery>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <RemoveAllIPforDiscoveryResponse xmlns="http://tempuri.org/">
      <RemoveAllIPforDiscoveryResult>
        <Status>Success or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
        <Message>string</Message>
      </RemoveAllIPforDiscoveryResult>
    </RemoveAllIPforDiscoveryResponse>
  </soap:Body>
</soap:Envelope>
```

<table>
<thead>
<tr>
<th>RemoveAllIPforDiscovery (sample)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;identityKey&gt;29efca900e186f3dd&lt;/identityKey&gt;</code></td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
</tbody>
</table>
RemoveIPforDiscovery

Remove the specific DeviceIPItem in the table discovery_device_by_ip in Postgresql database which are added by API AddIPforDiscovery.

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /workspace/1/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/RemoveIPforDiscovery"

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <RemoveIPforDiscovery xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <mgmtIP>string</mgmtIP>
    </RemoveIPforDiscovery>
  </soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <RemoveIPforDiscoveryResponse xmlns="http://tempuri.org/">
      <RemoveIPforDiscoveryResult>
        <Status>Success or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
        <Message>string</Message>
      </RemoveIPforDiscoveryResult>
    </RemoveIPforDiscoveryResponse>
  </soap:Body>
</soap:Envelope>
```
RemoveIPforDiscovery (sample)

<table>
<thead>
<tr>
<th>IdentityKey</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>29efca900e186f3dd</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
<tr>
<td>mgmtIP</td>
<td>Enter the IP address needed to be removed from Database. Type:string</td>
</tr>
</tbody>
</table>

CreateCustomizedDeviceAttribute

Creating new attribute for devices (assume an attribute can only be added on global basis and will be available for other devices as well).

Mode: Blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```xml
POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/CreateCustomizedDeviceAttribute"

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <CreateCustomizedDeviceAttribute xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <attributeName>string</attributeName>
    </CreateCustomizedDeviceAttribute>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <CreateCustomizedDeviceAttributeResponse xmlns="http://tempuri.org/">
      <CreateCustomizedDeviceAttributeResult>
        <Status>Success or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
        <Message>string</Message>
      </CreateCustomizedDeviceAttributeResult>
    </CreateCustomizedDeviceAttributeResponse>
  </soap:Body>
</soap:Envelope>
```
CreateCustomizedDeviceAttribute (sample) | Description
---|---
<identityKey> 29efc900e186f3dd</identityKey> | The unique key to identify the login session. Type: string.
<attributeName>Office</attributeName> | Add the customized attribute name to all the devices. Type: string.

**Note:** The user needs to execute API `RebuildL3Topology` to refresh the customized device attributes to workstation.

**SetCustomizedDeviceAttribute**

Set customized device attribute by device hostname and attribute name.

**Mode:** Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```xml
POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/SetCustomizedDeviceAttribute"

<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SetCustomizedDeviceAttribute xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <hostName>string</hostName>
      <attributeName>string</attributeName>
      <attributeValue>string</attributeValue>
    </SetCustomizedDeviceAttribute>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```
SetCustomizedDeviceAttribute (sample)

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;identityKey&gt;</code> 29efca90e186f3dd <code>&lt;identityKey&gt;</code></td>
</tr>
<tr>
<td><code>&lt;hostName&gt;</code> GW2Lab <code>&lt;hostName&gt;</code></td>
</tr>
<tr>
<td><code>&lt;attributeName&gt;</code> Office <code>&lt;attributeName&gt;</code></td>
</tr>
<tr>
<td><code>&lt;attributeValue&gt;</code> Boston <code>&lt;attributeValue&gt;</code></td>
</tr>
</tbody>
</table>

Note: The user needs to execute API RebuildL3Topology to refresh the customized device attributes to workstation.

GetCustomizedDeviceAttribute

Get customized device attribute by device hostname and attribute name.

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetCustomizedDeviceAttribute"

<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<GetCustomizedDeviceAttribute xmlns="http://tempuri.org/"/>
</soap:Body>
</soap:Envelope>
GetCustomizedDeviceAttribute (sample) | Description
---|---
<identityKey>29efca900e186f3dd</identityKey> | The unique key to identify the login session. Type: string.
<hostName>GW2Lab</hostName> | Input the hostname of the target device. Type: string.
<attributeName>Office</attributeName> | Specify the customized device attribute name. Type: string.

SetDeviceAttribute

Set device attribute value. Device hostname, attribute name and attribute value is required.

Mode: blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/SetDeviceAttribute"
SetDeviceAttribute (sample)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;identityKey&gt;</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
<tr>
<td>29efca900e186f3dd</td>
<td></td>
</tr>
<tr>
<td>&lt;hostName&gt;</td>
<td>Input the hostname of the target device. Type: string.</td>
</tr>
<tr>
<td>GW2Lab</td>
<td></td>
</tr>
<tr>
<td>&lt;attributeName&gt; Model</td>
<td>Specify the device attribute name, refer to Table 2 for all the device attributes of devices in NetBrain. Type: string.</td>
</tr>
<tr>
<td>&lt;attributeValue&gt; Model</td>
<td></td>
</tr>
<tr>
<td>2811</td>
<td>Set the value for the attribute. Type: string.</td>
</tr>
</tbody>
</table>

Note: The user needs to execute API RebuildL3Topology to refresh the device attributes to workstation.

GetDeviceAttribute

Extract device attribute out of NetBrain database.

Mode: Blocked
The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetDeviceAttribute"

<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetDeviceAttribute xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <hostName>string</hostName>
      <attributeName>string</attributeName>
    </GetDeviceAttribute>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetDeviceAttributeResponse xmlns="http://tempuri.org/">
      <GetDeviceAttributeResult>
        <Status>Success</Status> or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
        <Message>string</Message>
      </GetDeviceAttributeResult>
      <attributeValue>string</attributeValue>
    </GetDeviceAttributeResponse>
  </soap:Body>
</soap:Envelope>
```

**GetDeviceAttribute (sample)**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;identityKey&gt;</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
<tr>
<td>29efca900e186f3dd</td>
<td></td>
</tr>
<tr>
<td>&lt;hostName&gt;</td>
<td>Input the hostname of the target device. Type: string.</td>
</tr>
<tr>
<td>GW2Lab</td>
<td></td>
</tr>
<tr>
<td>&lt;attributeName&gt;</td>
<td>Specify the device attribute name, refer to Table 1 for all</td>
</tr>
<tr>
<td>Model</td>
<td>the device attributes of devices in NetBrain. Type: string.</td>
</tr>
</tbody>
</table>
Set Network Server of devices in the workspace.

Mode: Blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```xml
POST /workspace/1/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.123
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/SetNetWorkServerOfDevice"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SetNetWorkServerOfDevice xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <deviceHostName>string</deviceHostName>
      <networkServerName>string</networkServerName>
      <networkServerIP>string</networkServerIP>
    </SetNetWorkServerOfDevice>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SetNetWorkServerOfDeviceResponse xmlns="http://tempuri.org/">
      <SetNetWorkServerOfDeviceResult>
        <Status>Success or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
        <Message>string</Message>
      </SetNetWorkServerOfDeviceResult>
    </SetNetWorkServerOfDeviceResponse>
  </soap:Body>
</soap:Envelope>
```
<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;identityKey&gt; 29efca90e186f3dd/identityKey</td>
</tr>
<tr>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
<tr>
<td>&lt;deviceHostName&gt; GW2Lab/deviceHostName</td>
</tr>
<tr>
<td>Input the hostname of the target device. Type: string.</td>
</tr>
<tr>
<td>&lt;networkServerName&gt; NB-NS/networkServerName</td>
</tr>
<tr>
<td>Input the Network Server alias predefined in the Shared Network Settings. Type: string</td>
</tr>
<tr>
<td>&lt;networkServerIP&gt; 10.10.10.88/networkServerIP</td>
</tr>
<tr>
<td>Input the IP address or Domain name of the Network Server predefined in the Shared Network Settings. Type: string</td>
</tr>
</tbody>
</table>

**DeleteDeviceByHostName**

Delete device from NetBrain database and clean up all dependent data.

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/DeleteDeviceByHostName"

<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <DeleteDeviceByHostName xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <hostName>string</hostName>
    </DeleteDeviceByHostName>
  </soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Body>
    <DeleteDeviceByHostName xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <hostName>string</hostName>
    </DeleteDeviceByHostName>
  </soap:Body>
</soap:Envelope>
```
<DeleteDeviceByHostNameResponse xmlns="http://tempuri.org/">
    <DeleteDeviceByHostNameResult>
        <Status>Success</Status>
        <Message>string</Message>
    </DeleteDeviceByHostNameResult>
</DeleteDeviceByHostNameResponse>

<table>
<thead>
<tr>
<th>DeleteDeviceByHostName (sample)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;identityKey&gt; 29efca900e186f3dd&lt;/identityKey&gt;</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
<tr>
<td>&lt;hostName&gt; GW2Lab &lt;/hostName&gt;</td>
<td>Input the hostname of the device which is expected to remove from workspace. Type: string.</td>
</tr>
</tbody>
</table>

*Note: The user needs to execute API RebuildL3Topology to remove the devices from workstation.*

**RebuildL3Topology**

Rebuild L3 Topology after device import explicitly.

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```xml
POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/RebuildL3Topology"

<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <RebuildL3Topology xmlns="http://tempuri.org/">
            <identityKey>string</identityKey>
        </RebuildL3Topology>
    </soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RebuildL3TopologyResponse xmlns="http://tempuri.org/">
      <RebuildL3TopologyResult>
        <Status>Success</Status>
        <Message>string</Message>
      </RebuildL3TopologyResult>
    </RebuildL3TopologyResponse>
  </soap:Body>
</soap:Envelope>

<table>
<thead>
<tr>
<th>RebuildL3Topology (sample)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;identityKey&gt; 29efca900e186f3dd&lt;/identityKey&gt;</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
</tbody>
</table>

**SetDeviceSitePath**

Set device Site path by sitePath and deviceName.

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /workspace/1/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.247
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/SetDeviceSitePath"

<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SetDeviceSitePath xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
      <sitePath>string</sitePath>
      <deviceName>string</deviceName>
    </SetDeviceSitePath>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SetDeviceSitePathResponse xmlns="http://tempuri.org/">
      <SetDeviceSitePathResult>
        <Status>Success or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
        <Message>string</Message>
      </SetDeviceSitePathResult>
      </SetDeviceSitePathResponse>
    </soap:Body>
  </soap:Envelope>

SetDeviceSitePath (sample) | Description
---|---
<identityKey> 29efca900e186f3dd</identityKey> | The unique key to identify the login session. Type: string.
<sitePath> My Network\NB-LAB\Test</sitePath> | Specify the leaf site in the Site Tree. Type: string.
<deviceName>GW2Lab</deviceName> | Input the hostname of the device which is expected to set the site path. Type: string.

**Note:**
1. *The specified site must be built already and must be a Leaf Site.*
2. *For each API call of SetDeviceSitePath, NetBrain does not rebuild the site topology immediately. The user needs to call the API RebuildSiteTopology after all SetDeviceSitePath APIs are set.*

RebuildSiteTopology

Trigger Site topology calculation operations.

**Mode:** Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /workspace/1/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.247
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/RebuildSiteTopology"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RebuildSiteTopology xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
    </RebuildSiteTopology>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RebuildSiteTopologyResponse xmlns="http://tempuri.org/">
      <RebuildSiteTopologyResult>
        <Status>Success or Failed or Exception or ParseFileFailed or InvalidParam or InvalidLicense or OverMaxRecordFailed or PartSuccess</Status>
        <Message>string</Message>
      </RebuildSiteTopologyResult>
    </RebuildSiteTopologyResponse>
  </soap:Body>
</soap:Envelope>

<table>
<thead>
<tr>
<th>RebuildSiteTopology (sample)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;identityKey&gt; 29efca900e186f3dd&lt;/identityKey&gt;</td>
<td>The unique key to identify the login session. Type: string.</td>
</tr>
</tbody>
</table>

**Logout**

Mode: Not blocked

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /workspace/3/nbws3rdparty.asmx HTTP/1.1
Host: 10.10.10.52
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Logout"
Logout (sample)

```xml
<?xml version="1.0" encoding="utf-8"?>
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Logout xmlns="http://tempuri.org/">
      <identityKey>string</identityKey>
    </Logout>
  </soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

### Table 1

**Device Attributes for Soap API GetDeviceAttribute:**

<table>
<thead>
<tr>
<th>GetDeviceAttribute</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Device Property</strong></td>
<td></td>
</tr>
<tr>
<td>Vendor</td>
<td>The Vendor in the device properties.</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>The Model in the device properties.</td>
<td></td>
</tr>
<tr>
<td>SerialNumber</td>
<td>The Serial Number in the device properties.</td>
<td></td>
</tr>
<tr>
<td>SoftwareVersion</td>
<td>The Software Version in the device properties.</td>
<td></td>
</tr>
<tr>
<td>SystemMemory</td>
<td>The System Memory in the device properties.</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>The Location in the device properties.</td>
<td></td>
</tr>
<tr>
<td>Contact</td>
<td>The Description in the device properties.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>The Contact in the device properties.</td>
<td></td>
</tr>
</tbody>
</table>

| <identityKey> 29efca900e186f3dd</identityKey> | The unique key to identify the login session. Type: string. |
Device Setting

<table>
<thead>
<tr>
<th>AccessMethod</th>
<th>The method for accessing to devices: 512 → Access to device via SSH 1024 → Access to device via Telnet 8192 → Access to device via SSH Private key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnableUsernamePrompt</td>
<td>The privilege username prompt of devices. For instance, <strong>username:</strong></td>
</tr>
<tr>
<td>EnablePasswordPrompt</td>
<td>The privilege password prompt of devices. For instance, <strong>password:</strong></td>
</tr>
<tr>
<td>LoginPrompt</td>
<td>The no-privilege login prompt of devices. For instance, <strong>username:</strong></td>
</tr>
<tr>
<td>MgmtIP</td>
<td>The management IP of devices.</td>
</tr>
<tr>
<td>NonPrivilegePrompt</td>
<td>The non-privilege prompt of devices. For instance, <strong>GW2Lab&gt;</strong></td>
</tr>
<tr>
<td>PasswordPrompt</td>
<td>The non-privilege password prompt of devices. For instance, <strong>password:</strong></td>
</tr>
<tr>
<td>PrivilegePrompt</td>
<td>The privilege prompt of devices. For instance, <strong>GW2Lab#</strong></td>
</tr>
<tr>
<td>SshPort</td>
<td>The SSH port to devices. Default value 22.</td>
</tr>
<tr>
<td>SnmpPort</td>
<td>The SNMP port of devices.</td>
</tr>
<tr>
<td>SnmpVersion</td>
<td>The SNMP version of devices: 1 --&gt; SNMPv1 2 --&gt; SNMPv2c 3 --&gt; SNMPv3</td>
</tr>
<tr>
<td>TelnetPort</td>
<td>The telnet port to devices. Default value 23.</td>
</tr>
</tbody>
</table>

Table 2
Device Attributes for Soap API **SetDeviceAttribute:**

<table>
<thead>
<tr>
<th>SetDeviceAttribute</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vendor</strong></td>
<td></td>
<td>The Vendor in the device properties.</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td></td>
<td>The Model in the device properties.</td>
</tr>
<tr>
<td><strong>SerialNumber</strong></td>
<td></td>
<td>The Serial Number in the device properties.</td>
</tr>
<tr>
<td><strong>SoftwareVersion</strong></td>
<td></td>
<td>The Software Version in the device properties.</td>
</tr>
<tr>
<td><strong>SystemMemory</strong></td>
<td></td>
<td>The System Memory in the device properties.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td></td>
<td>The Location in the device properties.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
<td>The Description in the device properties.</td>
</tr>
<tr>
<td><strong>Contact</strong></td>
<td></td>
<td>The Contact in the device properties.</td>
</tr>
<tr>
<td><strong>AccessMethod</strong></td>
<td></td>
<td>The method for accessing to devices: 512 → Access to device via SSH 1024 → Access to device via Telnet 8192 → Access to device via SSH Private key</td>
</tr>
<tr>
<td><strong>EnableUsername</strong></td>
<td></td>
<td>The enable username to log to devices.</td>
</tr>
<tr>
<td><strong>EnablePassword</strong></td>
<td></td>
<td>The enable password to log to devices.</td>
</tr>
<tr>
<td><strong>MgmtIP</strong></td>
<td></td>
<td>The management IP of devices.</td>
</tr>
<tr>
<td><strong>SshPort</strong></td>
<td></td>
<td>The SSH port to devices. Default value 22.</td>
</tr>
<tr>
<td><strong>SnmpPort</strong></td>
<td></td>
<td>The SNMP port of devices.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>SnmpRo</td>
<td>The SNMP RO string of devices. It is encrypted by AES algorithm.</td>
<td></td>
</tr>
<tr>
<td>SnmpRw</td>
<td>The SNMP RW string of devices. It is encrypted by AES algorithm.</td>
<td></td>
</tr>
</tbody>
</table>
| SnmpVersion| The SNMP version of devices:  
1-->SNMPv1  
2-->SNMPv2c  
It does not support SNMPv3 in current release. |
| TelnetPort | The telnet port to devices. Default value 23.                               |
| UserName   | The username to log to devices.                                            |
| UserPassword| The password to log to devices.                                            |

2. **Remote Query of PostgreSQL Database**

2.1 **Introduction**

NetBrain uses PostgreSQL as the database to store the workspace data in the server, so it supports the query to the workspace data via standard SQL query. In this document we explain how to configure PostgreSQL so that it supports the remote query and provide the sample SQL scripts. The user can follow three steps: modify the configuration file; add the ready-only user for the database; run the sample SQL scripts against the databases which you would like to query.

2.2 **Configure PostgreSQL**

Browse through the database directory (the default is `<NetBrain Installation Directory>\Postgresql 8.4\database`). You can open services and applications Management console to find where the database is stored as shown in the following figure:

(The directory after –D is where PostgreSQL database installed)
1. **Modify postgresql.conf**

   Find the line
   
   ```
   #listen_addresses … ,
   ```
   
   and modify it to (The lines starting with # are comments):
   
   ```
   listen_addresses = '*'
   ```
   
   (Change this line to `listen_addresses = '*'`)

2. **Modify pg_hba.conf**

   Find the line
   
   ```
   # IPv4 local connections:
   ```
   
   And add the following lines after this line:
   
   ```
   host    all         all         0.0.0.0/0          md5
   ```
3. Restart the service PostgreSQL Database Server 8.4 for NetBrain.

2.3 Add the read only User to the database

1. From the start menu, open pgAdmin programming.
2. Create a new connection: please enter the port number, username and password you entered during the
3. Create a new login role
   Fill in the role name and password. Keep the default value for other fields.
4. Set this login role to be read only:
   Follow three steps shown in the following figure:

Copy and paste the following scripts into the window:

```sql
CREATE OR REPLACE FUNCTION adduserreadonly(username character varying)
RETURNS integer AS
$BODY$
declare
    r character varying(256);
BEGIN
    FOR r IN SELECT tablename FROM pg_tables WHERE tablename NOT LIKE 'pg%'
    AND tablename NOT LIKE 'sql_9%' ORDER BY tablename LOOP
        EXECUTE 'GRANT SELECT ON "'||r||'" TO '||username;
    END LOOP;
END;
$BODY$
```

END LOOP;

FOR r IN SELECT viewname FROM pg_views where schemaname='public' ORDER BY viewname LOOP
EXECUTE 'GRANT SELECT ON "'||r||'" TO '||username;
END LOOP;

return 1;
}
END;

$BODY$
LANGUAGE 'plpgsql' VOLATILE
COST 100;
ALTER FUNCTION adduserreadonly( character varying) OWNER TO postgres;

select adduserreadonly('user1'); --// Change "users" as the name of login role you created in the last step!
(Change "user1" as the name of login role you created in the last step! Click this button to run this script)

Close the window and select No
Now you can login to the database using the login role you just created. The database names are nbclici/workspace+integer and the default port is 54321.

Note: You may need to run the sql commands against all the databases which you would like to query from.

Try this SQL query:

```
select * from "user";
```

It should return a list of users.
2.4 Frequently used tables

The data is stored in the tables as shown above, here lists some useful tables. You can get more details by visiting the postgres in your server.

- **devicegroup**: Lists all device group info.
- **devices**: Lists all devices in your ES
- **devicesetting**: Lists the device setting info for each device.
- **lp2mac**: Lists all IP addresses with their corresponding MAC addresses.
- **lpphone**: Lists all IP phone info.
- **l2connectivity**: Lists L2 connectivities of your devices.
- **Nat**: Lists all nat configured devices
- **switchgroup**: Lists all your switch group info.
- **switchgroupdevice**: Lists all switches in your switch group.
- **user**: List all users in your ES

For more detailed information about tables, please refer to Appendix: **Asset Report Storage**

2.5 SQL Query Examples

- Get names of end systems in the device group “ppp”.
  
  Select devicename from devicegroupdeviceview where devicegroupname = 'ppp' and issubtype = 1004;
- Get the list of all devices.
  
  ```sql
  select * from devices;
  ```

- Get the device setting for device "xxx"
  
  ```sql
  select * from devicesetting where deviceid = (select id from devices where strname='xxx');
  ```

- Get the vendor model data for end systems
  
  ```sql
  SELECT stroid, strvendorname, strmodelname FROM system_vendormodel where idevicetype=1004;
  ```

- Get the names of end systems in the systemdevicegroup "#Island 1"
  
  ```sql
  select devicename from systemdevicegroupdeviceview where systemdevicegroupname='#Island 1' and isubtype=1004;
  ```

- Get all interface data for device with name containing the string “ENT”.
  
  ```sql
  select interfacename, interface_ip, description, devicename FROM interfacesettingview where devicename like '%ENT%';
  ```

- Get all devicespec data
  
  ```sql
  SELECT * FROM system_devicespec;
  ```

For more information about SQL query, please refer to Appendix: Asset Report Storage

### 2.6 Device Type

<table>
<thead>
<tr>
<th>Device ID</th>
<th>Device Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Cisco Router</td>
</tr>
<tr>
<td>102</td>
<td>Juniper Router</td>
</tr>
<tr>
<td>2021</td>
<td>Unclassified Router</td>
</tr>
<tr>
<td>2999</td>
<td>Cisco IOS XR</td>
</tr>
<tr>
<td>10417</td>
<td>Alcatel Lucent Service Router</td>
</tr>
<tr>
<td>2001</td>
<td>Cisco IOS Switch</td>
</tr>
<tr>
<td>2023</td>
<td>Extreme Switch</td>
</tr>
<tr>
<td>2041</td>
<td>HP Menu-Driven Switch</td>
</tr>
<tr>
<td>2011</td>
<td>HP ProCurve Switch</td>
</tr>
<tr>
<td>2012</td>
<td>Juniper EX Switch</td>
</tr>
<tr>
<td>2013</td>
<td>Arista Switch</td>
</tr>
<tr>
<td>2024</td>
<td>Brocade Switch</td>
</tr>
<tr>
<td>2015</td>
<td>Dell Force10 Switch</td>
</tr>
<tr>
<td>2110</td>
<td>Nortel Switch</td>
</tr>
<tr>
<td>2150</td>
<td>Enterasys Switch</td>
</tr>
<tr>
<td>3333</td>
<td>3Com Switch</td>
</tr>
<tr>
<td>2004</td>
<td>Cisco Nexus Switch</td>
</tr>
<tr>
<td>2061</td>
<td>Unclassified Switch</td>
</tr>
<tr>
<td>2140</td>
<td>Ciena Switch</td>
</tr>
<tr>
<td>10327</td>
<td>Dell Networking Switch</td>
</tr>
</tbody>
</table>
### 3. File-based inventory data synchronisation (HPNA integration)

- Import network data such as configurations and show command results from CM tools such as HP Network Automation (NA).

- CM tools export the data (configurations or show command results) into a shared file folder or a FTP server for NetBrain system to import.

- NetBrain benchmark can retrieve the data from a file folder via FTP service recursively.
3.1 Introduction

NetBrain discovery engine can help you discover the network quickly. However, if for some reasons your network does not allow NetBrain system to access the live network, NetBrain provides the function to import the network data from one specified folder on local disk or FTP site and then build L3 and L2 topology automatically. In this way, NetBrain System provides the integration ability with third-party systems such as HP Network Automation. This document will walk you through how to integrate with HP Network Automation (NA) step by step.

There are three major steps to integrate NetBrain OE system with HP NA system:

- Setup the schedule to retrieve the network data from HP NA. Create diagnostics in HP NA and set up a task to run diagnostics recursively. The diagnostic data is stored in HP database.
- Setup a task to import the diagnostic data into a file folder. NetBrain provides Perl script to do this.
- Setup a schedule discovery task in NetBrain workstation to import the data from the folder recursively. This discovery task should be scheduled to run after task 1.2 is done.

3.2 Define diagnostics to retrieve the data in HP NA

In this step, you will define diagnostics to retrieve the data from the network devices. The data includes configuration file, routing table, MAC table, CDP table, and ARP table.

a. Login HP Network Automation system interface and click Devices > Devices Tools > diagnostics to switch to the diagnostics view, click New diagnostics.

b. Refer to the table below to define the script and click save script to return to the command scripts page.

c. Select the script defined in the last step and click Run link on the right side to create a new task to run the script. Refer to the table below to define the task and then click the Save task button.
Repeat step a, b and c for other types of live data including route table, ARP table, MAC table and CDP table from live devices.

- For different device types, the commands to retrieve data may be different. So you need define different device groups and each group includes all devices using the same commands to retrieve the live data. Then for each device group, you define a diagnostic.
- All diagnostics results should be saved in the same directory.
- The following table lists show commands for different Cisco device types. For the commands to get VRF route table and ARP table, you need to replace the parameter $vrf_name with the actual VRF name.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cisco IOS Router</strong></td>
<td>show run</td>
</tr>
<tr>
<td></td>
<td>show ip route</td>
</tr>
<tr>
<td></td>
<td>show ip route vrf $vrf_name</td>
</tr>
<tr>
<td></td>
<td>show arp</td>
</tr>
<tr>
<td></td>
<td>show ip arp vrf $vrf_name</td>
</tr>
<tr>
<td></td>
<td>show cdp neighbor detail</td>
</tr>
<tr>
<td><strong>Cisco IOS XR</strong></td>
<td>show run</td>
</tr>
<tr>
<td></td>
<td>show ip route</td>
</tr>
<tr>
<td></td>
<td>show ip route vrf $VrfName</td>
</tr>
<tr>
<td></td>
<td>show cdp neighbor detail</td>
</tr>
<tr>
<td></td>
<td>show arp</td>
</tr>
<tr>
<td></td>
<td>show arp vrf $VrfName</td>
</tr>
<tr>
<td><strong>Cisco IOS switch (Layer 2)</strong></td>
<td>show run</td>
</tr>
<tr>
<td></td>
<td>show arp</td>
</tr>
<tr>
<td></td>
<td>show cdp neighbor detail</td>
</tr>
<tr>
<td></td>
<td>show mac-address-table</td>
</tr>
<tr>
<td></td>
<td>show mac address-table</td>
</tr>
</tbody>
</table>
Cisco IOS switch (Layer 3)

- `show run`
- `show ip route`
- `show ip route vrf $vrf_name`
- `show arp`
- `show ip arp vrf $vrf_name`
- `show cdp neighbor detail`
- `show mac-address-table`
- `show mac address-table`

Cisco Nexus Switch

- `show run all`
- `show ip route`
- `show ip route vrf $VrfName`
- `show cdp neighbor detail`
- `show ip arp`
- `show ip arp vrf $VrfName`
- `show mac-address-table`
- `show mac address-table`

Cisco Firewall

- `write terminal`
- `show route`

Note: For the other device types, you can get the show commands from corresponding tabs in the Device Driver. Under Workspace menu>Vendor Model dropdown list>Device Driver.

### 3.3 Define tasks to export live data from HP NA to a folder

In this step, you will create a task to run the script to export the diagnostic results into a file folder.

**a.** At HP NA interface click **devices > Device Tools > Command Script**. Click the **New command script** link.

**b.** In **New Command script** window, refer to the following table to define the script, then click **Save Script** button to return to the Command Scripts page.

<table>
<thead>
<tr>
<th>Name</th>
<th>Enter the name of the new script</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a description of the script</td>
</tr>
<tr>
<td>Script Type</td>
<td>Select General Purpose (the default)</td>
</tr>
<tr>
<td>Advanced Scripting</td>
<td>Checked Advanced scripting option</td>
</tr>
</tbody>
</table>

- **Device Family**: select Any Device Family option
- **Language**: select perl option
- **Parameters**: enter `$nb_file_dir$` and `$nb_showcmd_diagnostic$`, separated by a comma. The first parameter defines where the data is stored and should be the file folder. The second parameter is the name of diagnostics you created in step 1.1.
- **Script**: Copy and paste the following scripts.

```perl
#!/usr/bin/perl
use lib qw(blib lib);
use Getopt::Long;
```
use Pod::Usage;
use Opsware::NAS::Client;

my($host,$port,$user,$pass) = ('localhost','$tc_proxy_telnet_port$','$tc_user_username$','$tc_user_password$');
my $deviceID = '$tc_device_id$';
my $deviceName = '$tc_device_hostnames$';
if (!$deviceName) {
    $deviceName = '$tc_device_ids$';
}

# Get diagnostics file name
my $diagnostics = "$nb_showcmd_diagnotics$";
if (!$diagnostics) {
    print "Diagnostics name is needed.",
    exit(1);
}

# Open file
my $destDir = '$nb_file_dir$';
if (!$destDir) {
    $destDir = "c:\configs"
}
mkdir($destDir) unless (-e $destDir);

my $destFile = $destDir . $deviceName . ".configt$";
open(FILE, ">$destFile")|| die "Can't open $destFile: $!

print FILE "hostname 

my $nas = Opsware::NAS::Client->new();
my $login = $nas->login(-user => $user, -pass => $pass, -host => $host);
if (!$login->ok()) {
    print "Login failed.
    close(FILE);
    undef $nas;
    exit(1);
}

my $diag_list = $nas->list_diagnostic(diagnostic => $diagnostics, deviceID => $deviceID);
my $show;

    foreach my $diag ($diag_list->result()) {
        $show = $nas->show_diagnostic(id => $diag->DeviceDataID);
        # print all versions of the show command result
        # comment these if only print the lastest version
        print FILE $show->pop_result();
        print FILE "\n\n";
Pull Variables button

Click the Pull Variables button, set the value for parameters.

- **nb_file_dir** field: check **Limit Values To** and enter the directory name where you want to save the live data. For example, C:\ftp\showcmd\  
  **Note:** the last slash **CANNOT be missing**.

- **nb_showcmd_diagnostics** field: check **Limit Values To** and enter the diagnostics name defined in the step 1.1.

c. Select the script defined in the last step and click the Run link on the right to create a new task to run the script. Refer to the table below to define the task and then click **Save task**.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Name</td>
<td>Displays the Run Command Script name</td>
</tr>
<tr>
<td>Applies to</td>
<td>Select Single group option and select the device group item which the diagnostics should be applied to.</td>
</tr>
<tr>
<td>Start Date</td>
<td>Set the start time of the task</td>
</tr>
<tr>
<td>Task info</td>
<td>Keep the default settings</td>
</tr>
<tr>
<td>Pre-Task / Post-Task Snapshot Options</td>
<td>Keep the default settings</td>
</tr>
<tr>
<td>Scheduling Options</td>
<td>Set the scheduling parameters. <strong>You should select the same schedule option as the task to run diagnostic</strong> (step 1.1c) and make sure that the start time is set to the time after the diagnostic task has been done. If you are not sure how long the diagnostic task will take, please make an estimate and you can adjust later.</td>
</tr>
<tr>
<td>Task Logging</td>
<td>Keep the default settings</td>
</tr>
</tbody>
</table>

d. Repeat step a, b and c for other types of live data. The data should be stored in the same file folder. **Notice that the suffix should be changed with the live device data type and you need modify the script manually. For example, if you want to export the ARP table, you need search out the following line from the script.**

```perl
my $destFile = $destDir . $deviceName . "configt";
```

Then modify the string config to arp, the result is:
my $destFile = $destDir . $deviceName . ".arp";

The table below lists the relationship between live device data type and suffix.

<table>
<thead>
<tr>
<th>Live device data type</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration file</td>
<td>.config</td>
</tr>
<tr>
<td>ARP table</td>
<td>.arp</td>
</tr>
<tr>
<td>VRF ARP table</td>
<td>.vrfarp-vrfname</td>
</tr>
<tr>
<td>MAC table</td>
<td>.mac</td>
</tr>
<tr>
<td>CDP table</td>
<td>.cdp</td>
</tr>
<tr>
<td>routing table</td>
<td>.route</td>
</tr>
<tr>
<td>VRF route table</td>
<td>.vfrouting-vrfname</td>
</tr>
</tbody>
</table>

3.4 Schedule the schedule discovery task to import the data from the folder

In this step you setup a schedule discovery task at NetBrain workstation to import the data from the file folder. Login into NetBrain OE workstation, Launch the discovery window, follow the steps below to define the schedule discovery task.

1. Check the Enable Schedule option
2. Set the schedule option. You should select the same schedule option (e.g., once a week) as the task to run diagnostic (step 1.1c) and make sure that the start time is set to the time after the task at step 1.2c is done. If you are not sure, please make an estimate and you can adjust later.
3. Click the Add button in the Import CLI data from folder pane.
4. Click the browse button to select the folder in which the live data is saved
5. Click OK button
6. Check Rebuild L3 and L2 topology automatically option
7. Click OK button
Note:

1. Only the files satisfying the following rules will be imported during running schedule discovery
   - The file name must be the hostname of the devices existing in the workspace.
   - The suffix of configuration files must be "config". And the suffix of ARP table, ARP vrf table, CDP table, routing table, routing vrf table and mac-address table must be "arp", "vrfarp-vrfname", ".cdp", ".route", ".vrfrouting-vrfname" and ".mac".
2. If more than one folder is set, the system will import and parse the device data in turn during discovery.
4. Expose Dynamic Mapping to Third Party Tools (Netcool Integration)

4.1 Introduction

- NetBrain provides the command line interface to map the path between two points or map around a single device.

- Ticket systems such as Netcool and NetView can integrate with NetBrain via this command line interface. While the command line is invoked from an alert, the map will be created for this alert.

- With one click, a NetBrain map can be exported to a Visio or PNG file.

4.2 Setup Netcool Integration

1. Please download the NetBrain Netcool Integration Software from this link:

   http://forums.netbraintech.com/download/NetBrain_Netcool_Integration.zip

2. Exact the file and follow the instruction to install it.
Note: The system will prompt you to enter the Computer IP of OE workstation, if you also installed the OE in this server, please keep the default settings; otherwise, you may need to enter the IP address of the OE workstation.

3. In the netcool software, please set the correct parameters as shown below:

```
netbrainagent -d 10.10.20.1 -s 143.35.254.1 -m "hello, all"
```
There are three parameters:

- \( d \): destination IP address of a device, could be IP, hostname or IP/Mask
- \( s \): source IP address of a device, could be IP, hostname or IP/Mask
- \( m \): message to be displayed on Qmap, you can set alarm message here
  (Can be omitted)

After you execute the command, NetBrain will create a new Qmap, as shown below:
(You need to select a datafolder to calculate the path)

The path is calculated via cached datafolder, please click NetBrain logo at top left > Options > Others tab to set default datafolder, so all further calculation will via this datafolder.
**Calculate the path from the end system:** Define a server as the source to calculate the path. The Q-map is created to present the path from this server to the device having network problem.

In case you defined the source end system here, and if you also define the source using –s parameter, NetBrain will ignore that parameter and use the option here.

As currently we don't provide function to calculate the path via live network (we may consider to improve this in future release), after calculating the path via cached data folder, if you want to calculate the path via live network, click on the path, and NetBrain will fill the source/destination fields, change the discover settings via live network can calculate again.
5. Drive Third Party Tools from the Map

Users can add third party tools to NetBrain Workstation as external tools. The user can pass the device management IP address, telnet port and hostname as arguments.

- Launch Telnet/SSH tools such as SecureCRT and Putty
- Launch a web browser
- Launch any windows commands such as ping

Example: Add the Putty.exe application as an external tool:

- Click the Add button. A new item appears on the Menu Contents pane named {New Tool 1}.
- Select the {New Tool 1}. Modify the title to “Telnet via Putty”.
- Click the button. The Open window opens.
- Select the Putty.exe application file from the local computer and click the Open button.
- Select the -telnet s(ManagementIP) argument for the Putty application.
- Click the OK button.

6. Integrate with inventory management tools

- The inventory data (device and interface data) can be imported from CSV file to the NetBrain system.
- For the fields not recognized by NetBrain, the user can dynamically define a parameter from the Workstation and the customized parameters are saved in the server. The user can then import the data for these fields into NetBrain.
- The user can view all device or interface data through NetBrain property pane.
- In one click, the user can export the inventory to a CSV file from the Workstation.
Appendix: Asset Report Storage

The most common used tables by Asset Report are: devices, device_property, devicesetting, interfacesetting, module_property

1. Device Report

<table>
<thead>
<tr>
<th>Field</th>
<th>Table(column)</th>
<th>Sql Example</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>devices(strname)</td>
<td>select id from devices where strname = 's devicename';</td>
<td>Normally, we use device name to get ID, and then use this ID as a key to search in order tables.</td>
</tr>
<tr>
<td>Management IP</td>
<td>devicesetting(manageip)</td>
<td>select convertoip(manageip) from devicesetting</td>
<td>You can see this as a main example, since there are 2 steps:</td>
</tr>
<tr>
<td>Management Interface</td>
<td>device_property(ma nagement_interface)</td>
<td>select management_interface from device_property where deviceid in (select id from devices where strname = '$devicename');</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>1. Get $ID of a specific device. 2. Use the $ID in step 1 to query target table.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device Type</td>
<td>devices(isubtype)</td>
<td>select isubtype from devices where devicename = '$devicename';</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This constant is internally used by NetBrain to stand for different device types. Such as 2 for Cisco Routers, 102 for Juniper Routers. You can get the corresponding relation in resource\DeviceType. e.g. You want to know number 2 stand for what, you can find in resource\DeviceType\2\main.xml to get the detail type info.</td>
<td></td>
</tr>
<tr>
<td>sysObjectID</td>
<td>device_property(sysobjectid)</td>
<td>select sysobjectid from device_property where deviceid in (select id from devices where strname = '$devicename');</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Vendor</td>
<td>device_property(vendor)</td>
<td>select vendor from device_property where deviceid in (select id from devices where strname = '$devicename');</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>device_property(model)</td>
<td>select model from device_property</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Software Version</strong></td>
<td>device_property(software_version)</td>
<td>select software_version from device_property where deviceid in (select id from devices where strname = 'device_name');</td>
<td>...</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td><strong>Serial Number</strong></td>
<td>device_property(serial_number)</td>
<td>select serial_number from device_property where deviceid in (select id from devices where strname = 'device_name');</td>
<td>...</td>
</tr>
<tr>
<td><strong>MAC Address</strong></td>
<td>device_property(manageif_mac)</td>
<td>select manageif_mac from device_property where deviceid in (select id from devices where strname = 'device_name');</td>
<td>...</td>
</tr>
<tr>
<td><strong>Group</strong></td>
<td>devices, devicegroup, devicegroupdevice</td>
<td>select strname from devicegroup where id in (select devicegroupid from devicegroupdevice where deviceid in (select id from devices where strname = 'device_name'));</td>
<td>There are 3 tables to store the corresponding information. Normally, 1. Get $DeviceID in table devices. 2. Get $GroupID in table devicegroupdevice. 3. Get $GroupName in table devicegroup</td>
</tr>
<tr>
<td><strong>Primary Controller</strong></td>
<td>lwap(primarycontroller)</td>
<td>select primarycontroller from lwap where</td>
<td>Only works for LWAP device.</td>
</tr>
<tr>
<td>Feature</td>
<td>Query</td>
<td>Note</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------</td>
<td></td>
</tr>
<tr>
<td>Secondary Controller</td>
<td><code>lwap(secondarycontroller)</code> select secondarycontroller from lwap where hostname = '$devicename'</td>
<td>Only works for LWAP device.</td>
<td></td>
</tr>
<tr>
<td>Default Gateway</td>
<td><code>lwap(defaultgateway)</code> select defaultgateway from lwap where hostname = '$devicename'</td>
<td>Only works for LWAP device.</td>
<td></td>
</tr>
<tr>
<td>Asset Tag</td>
<td><code>device_property(asset_tag)</code> select asset_tag from device_property where deviceid in (select id from devices where strname = '$devicename');</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>System Memory</td>
<td><code>device_property(system_memory)</code> select system_memory from device_property where deviceid in (select id from devices where strname = '$devicename');</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td><code>device_property(location)</code> select location from device_property where deviceid in (select id from devices where strname = '$devicename');</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Contact</td>
<td><code>device_property(contact)</code> select contact from device_property where deviceid in (select id from devices where strname = '$devicename');</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>In Site</td>
<td><code>devices, sitecluster2device, site, sitecluster,</code> select name from site where id in (</td>
<td>1. Get device belonging site $ID or $name using table devices, sitecluster2device, site.</td>
<td></td>
</tr>
</tbody>
</table>
### NetBrain System Integration Guide

**Table:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Table(column)</th>
<th>Sql Example</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device Name</strong></td>
<td>devices(strname)</td>
<td>select id from devices where strname = '$devicename';</td>
<td></td>
</tr>
<tr>
<td><strong>First Discovered At</strong></td>
<td>discover_newdevice(findtime)</td>
<td>select findtime from discover_newdevice where hostname = '$devicename';</td>
<td></td>
</tr>
<tr>
<td><strong>Last Discovered At</strong></td>
<td>discover_newdevice(lasttime)</td>
<td>select lasttime from discover_newdevice where hostname = '$devicename';</td>
<td></td>
</tr>
</tbody>
</table>

**2. Interface Report**

<table>
<thead>
<tr>
<th>Field</th>
<th>Table(column)</th>
<th>Sql Example</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device Name</strong></td>
<td>devices(strname)</td>
<td>select id from devices where strname = '$devicename';</td>
<td></td>
</tr>
<tr>
<td>Interface Name</td>
<td>Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><code>interfacesetting(interfacename)</code> or <code>interfacesetting(interfacefullname)</code></td>
<td><code>select interfacename from interfacesetting where (interfacename = '$interfacename' and deviceid in (select id from devices where strname = '$devicename'));</code></td>
<td>The $interfacename is a internally expression by NetBrain since the SNMP, Live Config or All kinds of tables(MAC, ARP, CDP…) may have different name of a same interface, we internally translate all the interface to a single expression. The $interfacefullname is the interface name shown up in config file.</td>
<td></td>
</tr>
<tr>
<td>Interface IP</td>
<td><code>interfacesetting(interface_ip)</code></td>
<td><code>select interface_ip from interfacesetting where (interfacename = '$interfacename' and deviceid in (select id from devices where strname = '$devicename'));</code></td>
<td>Normally, we use a device name and interface name to get a interface corresponding information</td>
</tr>
<tr>
<td>MIB Index</td>
<td><code>interfacesetting(mibindex)</code></td>
<td><code>select mibindex from interfacesetting where (interfacename = '$interfacename' and deviceid in (select id from devices where strname = '$devicename'));</code></td>
<td>...</td>
</tr>
<tr>
<td>Bandwidth</td>
<td><code>interfacesetting(bandwidth)</code></td>
<td><code>select bandwidth from interfacesetting where (interfacename = '$interfacename' and deviceid in (select id from devices where ...)</code></td>
<td>...</td>
</tr>
<tr>
<td><strong>NetBrain System Integration Guide</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>
| **strname** = \$devicename\');
| **Speed** | interfacesetting(speed_int) | select speed_int from interfacesetting where ( interfacename = \$interfacename\' and deviceid in ( select id from devices where strname = \$devicename\') );
| **Duplex** | interfacesetting(duplex) | select duplex from interfacesetting where ( interfacename = \$interfacename\' and deviceid in ( select id from devices where strname = \$devicename\') );
| **Live Status** | interfacesetting(interface_status) | select interface_status from interfacesetting where ( interfacename = \$interfacename\' and deviceid in ( select id from devices where strname = \$devicename\') );
<p>| <strong>MAC Address</strong> | interfacesetting(mac_address) | select mac_address from interfacesetting where ( interfacename = $interfacename' and deviceid in ( |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>SQL Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slot#</td>
<td>select module_slot from interfacesetting</td>
<td>select module_slot from interfacesetting where (interfacename = '$interfacename' and deviceid in (select id from devices where strname = '$devicename'))</td>
</tr>
<tr>
<td>Module Type</td>
<td>select module_type from interfacesetting</td>
<td>select module_type from interfacesetting where (interfacename = '$interfacename' and deviceid in (select id from devices where strname = '$devicename'))</td>
</tr>
<tr>
<td>Description</td>
<td>select description from interfacesetting</td>
<td>select description from interfacesetting where (interfacename = '$interfacename' and deviceid in (select id from devices where strname = '$devicename'))</td>
</tr>
<tr>
<td>Field1</td>
<td>The customized interface info, normally rarely used.</td>
<td>Field1</td>
</tr>
<tr>
<td>Field2</td>
<td></td>
<td>Field2</td>
</tr>
<tr>
<td>Field3</td>
<td></td>
<td>Field3</td>
</tr>
</tbody>
</table>
3. Module Report

<table>
<thead>
<tr>
<th>Field</th>
<th>Table(column)</th>
<th>Sql Example</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Name</td>
<td>devices(strname)</td>
<td>select id from devices where strname = '$devicename';</td>
<td>...</td>
</tr>
<tr>
<td>Slot</td>
<td>module_property(slot)</td>
<td>select slot from module_property where deviceid in (</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>select id from devices where strname = '$devicename';</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>);</td>
<td></td>
</tr>
<tr>
<td>Module Type</td>
<td>module_property(card_type)</td>
<td>select card_type from module_property where deviceid in (</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>select id from devices where strname = '$devicename';</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>);</td>
<td></td>
</tr>
<tr>
<td>Ports</td>
<td>module_property(ports)</td>
<td>select ports from module_property where deviceid in (</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>select id from devices where strname = '$devicename';</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>);</td>
<td></td>
</tr>
<tr>
<td>Serial Number</td>
<td>module_property(serial_number)</td>
<td>select serial_number from module_property where deviceid in (</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>select id from devices where strname = '$devicename';</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>);</td>
<td></td>
</tr>
<tr>
<td>HW Rev</td>
<td>module_property(hwrev)</td>
<td>select hwrev from module_property where deviceid in (</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>select id from devices where strname = '$devicename';</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>);</td>
<td></td>
</tr>
<tr>
<td>FW Rev</td>
<td>module_property(fwrev)</td>
<td>select fwrev from module_property where deviceid in (</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>);</td>
<td></td>
</tr>
</tbody>
</table>
4. VSS Report

See Module Report

5. Stackable Switch Report

See Module Report

7. Site Report

<table>
<thead>
<tr>
<th>Field</th>
<th>Table(column)</th>
<th>Sql Example</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>site(name)</td>
<td>select name from site where name = '$sitename';</td>
<td>...</td>
</tr>
<tr>
<td>Region</td>
<td>site(region)</td>
<td>select region from site where name = '$sitename';</td>
<td>...</td>
</tr>
<tr>
<td>Field</td>
<td>SQL Query</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Location/Address</td>
<td><code>site(location_address)</code> select location_address from site where name = 'ssitename';</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Employee Number</td>
<td><code>site(employee_number)</code> select employee_number from site where name = 'ssitename';</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device Count</td>
<td><code>sitecluster2device</code> select count(siteclusterid) from sitecluster2device where siteclusterid in( select id from site where name = 'ssitename');</td>
<td>We don't have a field to store this. You can calculate devices rows to get the numbers.</td>
<td></td>
</tr>
<tr>
<td>Contact Name</td>
<td><code>site(contact_name)</code> select contact_name from site where name = 'ssitename';</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone Number</td>
<td><code>site(phone_number)</code> select phone_number from site where name = 'ssitename';</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td><code>site(email)</code> select email from site where name = 'ssitename';</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td><code>site(Type)</code> select Type from site where name = 'ssitename';</td>
<td>NetBrain consider site into 3 major categories: Container Site, Leaf Site, and Cluster.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td><code>site(description)</code> select description from site where name = 'ssitename';</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field1</td>
<td>...</td>
<td>The customized module info, normally rarely used.</td>
<td></td>
</tr>
<tr>
<td>Field2</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field3</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>