



Data Extraction Engine

CiscoWorks Campus Manager Data Extraction Engine (DEE) is a utility to export Campus Manager application data.

This utility provides servlet and command line access to Campus Manager application data (User Tracking, Topology and Discrepancy), and allows you to extract data in Extensible Markup Language (XML) format.

This chapter contains:

- [Overview of Data Extraction Engine, page 14-2](#)
- [The cmexport Command, page 14-3](#)
- [cmexport User Tracking, page 14-8](#)
- [cmexport Topology Command, page 14-12](#)
- [cmexport Discrepancy Command, page 14-15](#)
- [cmexport Manpage, page 14-18](#)
- [DEE Developer's Reference, page 14-21](#)
- [Frequently Asked Questions, page 14-39](#)

Overview of Data Extraction Engine

Data Extraction Engine (DEE) is a utility that provides servlet access to User Tracking, Layer 2 topology, and discrepancy data.

It also includes a command line utility that you can use to fetch user tracking, Layer 2 topology, and discrepancy data for devices discovered by Campus Manager server.

This utility supports the following features:

- Generating user tracking data in XML format:

Allows you to access servlet and command line utilities that can generate user tracking data for devices discovered by Campus Manager Server.

- Generating Layer 2 topology data in XML format:

Allows you to generate the latest Layer 2 topology data including information on neighbor devices. Elements in XML file are created at the device level.

- Generating discrepancy data in XML format:

Allows you to use Campus Manager discrepancy APIs to retrieve latest discrepancy data from Campus Manager Server.

- Archiving XML Data:

Data generated through CLI is archived at the following locations:

Table 14-1 Data Archive Locations

For...	Location
User Tracking	PX_DATADIR/cmexport/ut/timestamput.xml
Layer 2 Topology	PX_DATADIR/cmexport/L2Topology/timestampL2Topology.xml
Discrepancy	PX_DATADIR/cmexport/Discrepancy/timestampDiscrepancy.xml

where PX_DATADIR is either %NMSROOT%/files folder (on Windows) or /var/adm/CSCOpX/files directory (on Solaris).

NMSROOT is the directory where you installed Campus Manager; timestamp is the time at which the log was written in *YearMonthDateHourOfDayMinuteSecond* format.

You can also specify a directory to store the output. This utility does not delete the files created in the archive. You should delete these files when necessary. While generating data through the servlet, the output appears at the client terminal.

- Generating user tracking and configuration data in XML format using the Servlet:

Allows you to generate and download the user tracking, topology and discrepancy XML files using the servlet.

You must upload a payload XML file, which contains the **cmexport** and **utexport** command options and CiscoWorks user credentials.

You should write your own script to invoke the servlet with a payload of this XML file. If the credentials are correct and options are valid, the servlet returns the exported file in XML format.

The cmexport Command

cmexport is the CiscoWorks Campus Manager command line interface for exporting discrepancy and Layer 2 topology data details into XML format.

Running cmexport Command

This section contains:

- [Command Line Syntax, page 14-3](#)
- [Commands, page 14-4](#)

Command Line Syntax

The command line syntax of the utility is in the following format:

```
cmexport command arguments options
```

where:

- `cmexport` is the CiscoWorks Campus Manager command line interface for exporting User Tracking, Layer 2 topology, and discrepancy data details into XML format.
- *command* specifies which core operation is to be performed.
- *arguments* are the additional parameters required for each core command.
- *options* are the optional parameters, which modify the behavior of the specific DEE core command.

The order of the arguments and options are not important. However, you must enter the core command immediately after `cmexport`.

Commands

[Table 14-2](#) lists the command part of the `cmexport` syntax.

Table 14-2 *Command Descriptions*

Core Command	Description
<code>ut</code>	Generates User Tracking data in XML format.
<code>l2topology</code>	Generates layer 2 topology data in XML format.
<code>discrepancy</code>	Generates discrepancy data in XML format.

You must invoke the `cmexport` command with one of the core commands specified in the above table. If you do not specify any core commands, `cmexport` can only execute the `-v` or `-h` options:

- Option `-v` displays the version of the `cmexport` utility
- Option `-h` (or null option) lists the usage information for this utility.

Arguments and Options

This section contains:

- [Mandatory Arguments, page 14-5](#)
- [Optional Arguments, page 14-6](#)
- [Function-Specific Options, page 14-6](#)

Mandatory Arguments

The argument that must be specified with all functions is:

- `-u userid`: Specifies the CiscoWorks userid.
- `-p password`: Specifies the password for CiscoWorks userid.

If you want to avoid the `-p` option which will reveal the password in clear text in CLI, you must store your userid and password in a file and set a variable `CMEXPORTFILE` which points to this file.

You must maintain this file and control access permissions to prevent unauthorized access. `cmexport` looks for current working directory if `CMEXPORTFILE` is set only to the file name instead of to the full path.

If you use the `-p` option, even after setting the `CMEXPORTFILE` variable, the password is taken from the command line instead of from `CMEXPORTFILE`. This is not secure and we recommend that you do not use this option.

You must enter the password in the file in the following format:

userid password

where *userid* is the CiscoWorks user name given in the command line. The delimiter between the userid and *password* is single blank space.

You must provide the delimiter if the password is blank. Otherwise, `cmexport` will not validate the password.

The password file can contain multiple entries with different user names. If there are duplicate entries the password that matches the first user name is considered.



Note

If `-p password` is used, the password is read from the command line instead of `CMEXPORTFILE`. This is not secure and we recommend that you do not use this option.

Optional Arguments

The arguments you can specify with any function are:

- `-a debuglevel`

Sets the debug level based on which debug information is printed. There are two levels of debugging—TRACE and DEBUG. If you do not specify the `-d` option, no logging will occur.

- `-l logfile`

Logs the results of the `cmexport` command to the specified log file name. By default the command output is displayed in the standard output.

Function-Specific Options

DEE supports the following function-specific option:

`-f filename`

If used with:

- User Tracking function

Specifies the name of the file to which the user tracking information is to be exported.

- Topology function

Specifies the name of the file to which the layer 2 topology information is to be exported.

- Discrepancy function

Specifies the name of the file to which the discrepancy information is to be exported.

Accessing Help

Enter the following at a CLI:

- `cmexport -h`

Displays a list of options for `cmexport`.

- `cmexport -h`

Displays a list of options for the `cmexport` command.

On Solaris, you can also enter the following at a CLI:

```
man cmexport
```

Uses of `cmexport`

If you enter:

```
cmexport ut {-u userid} -p password -host -f filename.xml
```

User Tracking XML output for host will be generated and it is stored in the file *filename.xml*.

If you want to export the latest topology details for all Layer 2 devices enter:

```
cmexport L2Topology {-u userid} -p password -f filename.xml
```

If you want to export the latest discrepancy details, enter:

```
cmexport Discrepancy {-u userid} -p password -f filename.xml
```

Notations

The notations followed in describing the command line arguments are explained below:

{argument}—Argument is a mandatory parameter.

[argument]—Argument is an optional parameter.

argument—Argument is a variable.

argument 1 | argument 2—Either argument 1 or argument 2 may be specified but not both.

[Table 14-3](#) lists the notations part of the `cmexport` syntax.

Table 14-3 Notations Descriptions

Command	Description
<code>ut</code>	<code>cmexport ut {-u <i>userid</i>} [-p <i>password</i>] -host [<i>host-options</i>] -phone [<i>phone-options</i>] [<i>options</i>]</code>
<code>l2topology</code>	<code>{ -u <i>userid</i> } [-p <i>password</i>] [-f <i>filename</i>]</code>
<code>discrepancy</code>	<code>{ -u <i>userid</i> } [-p <i>password</i>] [-f <i>filename</i>]</code>
<code>empty</code>	<code>[-v -h]</code>

`-v`—Displays the version of the `cmexport` utility.

`-h`—Lists the options available and function of each option.

cmexport User Tracking

This topic describes the `cmexport` User Tracking command, and the various options available to you. It contains the following sections:

- [Name, page 14-8](#)
- [Synopsis, page 14-8](#)
- [Description, page 14-9](#)
- [Mandatory Arguments, page 14-9](#)
- [Accessing Help, page 14-11](#)
- [Examples, page 14-11](#)

Name

`cmexport ut`: CiscoWorks `cmexport` user tracking function

Synopsis

```
cmexport ut: {-u userid} [-p password] -host [ host-options ] | -phone [ phone-options ] [ options ]
```

Table 14-4 lists the command part of the `cmexport` syntax.

Table 14-4 Command Descriptions

Argument	Can be one of the Following...
host-options	-query <i>queryname</i> -query <i>queryname</i> -view <i>viewname</i> -layout <i>layoutname</i> -layout <i>layoutname</i> -view <i>viewname</i> -query <i>queryname</i> -layout <i>layoutname</i> -query <i>queryname</i> -layout <i>layoutname</i> -view <i>viewname</i>
phone-options	-queryPhone <i>queryname</i> -layoutPhone <i>layoutname</i> -queryPhone <i>queryname</i> -layoutPhone <i>layoutname</i>
options	-f <i>filename</i> -d <i>debuglevel</i> -l <i>logfile</i>

Description

User Tracking (specified by `ut`) exports the user tracking data into an XML file based on a predefined schema.

Mandatory Arguments

The options that must be specified with the `cmexport ut` function are:

- **-u** *userid*: Specifies the CiscoWorks userid.
- **-p** *password*: Specifies the password for CiscoWorks userid.

If you want to avoid **-p** option which will reveal the password in clear text in CLI, you must store your userid and password in a file and set a variable `CMEXPORTFILE` which points to this file.

You must maintain this file and control access permissions to prevent unauthorized access. `cmexport` looks for current working directory if `CMEXPORTFILE` is set only to the file name instead of to the full path.

If you use the **-p** option, even after setting the *CMEXPORTFILE* variable, the password is taken from the command line instead of from *CMEXPORTFILE*. This is not secure and we recommend that you do not use this option.

The password must be provided in the file in the following format:

userid password

where *userid* is the CiscoWorks user name given in the command line. The delimiter between the *userid* and password is single blank space.

You must provide the delimiter if the password is blank. Otherwise, **cmexport** will not validate the password. The password file can contain multiple entries with different user names. The password that matches the first user name is considered in case of duplicate entries.



Note If **-p password** is used, the password is read from the command line instead of *CMEXPORTFILE*. This is not secure and we recommend that you do not use this option.

- **-host**: Specifies host data to be exported.
- **-phone**: Specifies phone data to be exported.

Options

The options you can specify with the **ut** function are:

- **-d debuglevel**

Sets the debug level based on which debug information is printed. There are two levels of debugging—TRACE and DEBUG. If you do not specify the **-d** option, no logging will occur.

- **-l logfile**

Logs the results of the **cmexport** command to the specified logfile name. By default the command output will be displayed in the standard output.

- **-f filename**

The file option specifies the filename where the XML output is to be stored. If the filename is not specified with **-f** option, an XML file of the format `timestamput.xml` is stored in the following directory:
PX_DATADIR/cmexport/ut

- **-view**
Specifies the format in which the user tracking XML data is to be presented. It supports two optional arguments:
 - a. **switch**: User Tracking data are displayed based on the type of switch.
 - b. **subnet**: User Tracking data are displayed based on subnet in which they are present.The **-view** options are not case sensitive.
- **-query** *queryname*
User Tracking host data is exported in XML format for the query provided in *queryname*. This option must be used with the **-host** argument.
- **-layout** *layoutname*
User Tracking host data is exported in XML format for the layout provided in *layoutname*. This option must be used with the **-host** argument.
- **-queryPhone** *queryname*
User Tracking phone data is exported in XML format for the query given in *queryname*. This option must be used with the **-phone** argument.
- **-layoutPhone** *layoutPhone*
User Tracking phone data is exported in XML format for the layout given in *layoutPhone*. This option must be used with the **-phone** argument.

Accessing Help

Enter the following at a CLI:

- `cmexport -h`: Displays a list of options for `cmexport`.
- `cmexport ut -h`: Displays a list of options for the `cmexport ut` command.

On Solaris, you can also enter the following at a CLI:

```
man cmexport
```

Examples

Considering `userid: admin`, `password: admin`, `queryname: host1Query`, `layoutname: host1Layout`, `queryphone: phone1Query`, `layoutphone: phone1Layout`, `filename: file1.xml`, we can have the following:

```

cmexport ut -u admin -p admin -host
cmexport ut -u admin -p admin -phone
cmexport ut -u admin -p admin -host -query host1Query -layout all
cmexport ut -u admin -p admin -host -query host1Query -layout
layoutname
cmexport ut -u admin -p admin -phone -queryPhone phone1Query
-layoutPhone phone1Layout
cmexport ut -u admin -p admin -host -f file1.xml
cmexport ut -u admin -view switch -host

```

cmexport Topology Command

This section contains:

- [Name, page 14-12](#)
- [Synopsis, page 14-12](#)
- [Description, page 14-13](#)
- [Mandatory Arguments, page 14-13](#)
- [Accessing Help, page 14-14](#)
- [Examples, page 14-14](#)

Name

`cmexport L2Topology`: CiscoWorks `cmexport` layer 2 topology function

Synopsis

```
cmexport l2topology { -u userid } [ -p password ] [ options ]
```

Table 14-5 *Command Description*

argument	can be one of the following...
options	<ul style="list-style-type: none"> -f <i>filename</i> -d <i>debuglevel</i> -l <i>logfile</i>

where `cmexport l2topology -h` lists the options available and function of each option.

Description

Layer 2 Topology (specified by `l2topology`) exports the Layer 2 topology data into an XML file based on a predefined schema.

Mandatory Arguments

The options that you must specify with the `cmexport L2Topology` function are:

- **-u** *userid*: Specifies the CiscoWorks userid.
- **-p** *password*

Specifies the password for CiscoWorks userid.

If you want to avoid **-p** option which will reveal the password in clear text in CLI, you must store your userid and password in a file and set a variable `CMEXPORTFILE` which points to this file.

You must maintain this file and control access permissions to prevent unauthorized access. `cmexport` looks for current working directory if `CMEXPORTFILE` is set only to the file name instead of to the full path.

If you use the **-p** option, even after setting the `CMEXPORTFILE` variable, the password is taken from the command line instead of from `CMEXPORTFILE`. This is not secure and we recommend that you do not use this option.

The password must be provided in the file in the following format:

userid password

where *userid* is the CiscoWorks user name given in the command line. The delimiter between the *userid* and *password* is single blank space.

You must provide the delimiter if the password is blank. Otherwise, `cmexport` will not validate the password. The password file can contain multiple entries with different user names. The password that matches the first user name is considered in case of duplicate entries.



Note If **-p** *password* is used, the password is read from the command line instead of `CMEXPORTFILE`. This is not secure and we recommend that you do not use this option.

Options

The options you can specify with the layer 2 topology function are:

- **-d** *debuglevel*

Sets the debug level based on which debug information is printed. There are two levels of debugging—TRACE and DEBUG. If you do not specify the **-d** option, no logging will occur.

- **-l** *logfile*

Logs the results of the **cmexport** command to the specified logfile name. By default the command output will be displayed in the standard output.

- **-f** *filename*

The file option specifies the filename where the XML output is to be stored. If the filename is not specified with **-f** option an XML file of the format `timestampL2Topology.xml` is stored in the following directory:
PX_DATADIR/cmexport/L2Topology

Accessing Help

Enter the following at a CLI:

```
cmexport -h: Displays a list of options for cmexport.
```

```
cmexport l2topology -h: Displays a list of options for the cmexport l2topology command.
```

On Solaris, you can also enter the following at a CLI:

```
man cmexport
```

Examples

Considering userid: admin, password: admin, filename: file1.xml, you can have the following:

```
cmexport L2Topology -u admin -p admin
cmexport L2Topology -u admin -p admin -f file1.xml
cmexport L2Topology -u admin -l file.log
```

cmexport Discrepancy Command

This section contains:

- [Name](#), page 14-15
- [Synopsis](#), page 14-15
- [Description](#), page 14-15
- [Mandatory Arguments](#), page 14-16
- [Accessing Help](#), page 14-17
- [Examples](#), page 14-17

Name

`cmexport Discrepancy`: CiscoWorks `cmexport Discrepancy` function.

Synopsis

`cmexport discrepancy` {**-u** *userid*} [**-p** *password*] [**options**]

where

Table 14-6 *Command Description*

Argument	Can be one of the Following
options	-f <i>filename</i> -d <i>debuglevel</i> -l <i>logfile</i>

`cmexport discrepancy -help` lists the options available and function of each option.

Description

Discrepancy (specified by Discrepancy) exports the Discrepancy data into an XML file based on a predefined schema.

Mandatory Arguments

The options that you must specify with the `cmexport` Discrepancy function are:

- `-u userid`: Specifies the CiscoWorks userid.
- `-p password`

Specifies the password for CiscoWorks userid.

If you want to avoid `-p` option which will reveal the password in clear text in CLI, you must store your userid and password in a file and set a variable `CMEXPORTFILE` which points to this file.

You must maintain this file and control access permissions to prevent unauthorized access. `cmexport` looks for current working directory if `CMEXPORTFILE` is set only to the file name instead of to the full path.

If you use the `-p` option, even after setting the `CMEXPORTFILE` variable, the password is taken from the command line instead of from `CMEXPORTFILE`. This is not secure and we recommend that you do not use this option.

The password must be provided in the file in the following format:

userid password

where *userid* is the CiscoWorks user name given in the command line. The delimiter between the userid and password is single blank space.

You must provide the delimiter if the password is blank. Otherwise, `cmexport` will not validate the password. The password file can contain multiple entries with different user names. The password that matches the first user name is considered in case of duplicate entries.



Note

If `-p password` is used, the password is read from the command line instead of `CMEXPORTFILE`. This is not secure and we recommend that you do not use this option.

Options

The options you can specify with the Discrepancy function are:

- *-a debuglevel*

Sets the debug level based on which debug information is printed. There are two levels of debugging—TRACE and DEBUG. If you do not specify the *-d* option, no logging will occur.

- *-l logfile*

Logs the results of the `cmexport` command to the specified log file name. By default the command output will be displayed in the standard output.

- *-f filename*

The file option specifies the filename where the XML output is to be stored. If the filename is not specified with *-f* option an XML file of the format `timestampDiscrepancy.xml` is stored in the following directory:
PX_DATADIR/cmexport/Discrepancy

Accessing Help

Enter the following at a CLI:

```
cmexport -h: Displays a list of options for cmexport.
```

```
cmexport discrepancy -h: Displays a list of options for the cmexport  
discrepancy command.
```

On Solaris, you can also enter the following at a CLI:

```
man cmexport
```

Examples

Considering `userid: admin, password:admin, filename: file1.xml`, you can have the following:

```
cmexport Discrepancy -u admin -p admin  
cmexport Discrepancy -u admin -p admin -f file1.xml  
cmexport Discrepancy -u admin -d 2
```

cmexport Manpage

This sections contains:

- [Command Line Syntax, page 14-18](#)
- [Commands, page 14-18](#)
- [Arguments and Options, page 14-19](#)
- [Accessing Help, page 14-21](#)

Command Line Syntax

The command line syntax of the utility is in the following format:

`cmexport command arguments options`

where:

- `cmexport` is the CiscoWorks Campus Manager command line interface for exporting User Tracking, Layer 2 topology, and discrepancy data details into XML format.
- `command` specifies which core operation is to be performed.
- `arguments` are the additional parameters required for each core command.
- `options` are the optional parameters, which modify the behavior of the specific DEE core command.

The order of the arguments and options is not important. However, you must enter the core command immediately after `cmexport`.

Commands

[Table 14-7](#) lists the command part of the `cmexport` syntax.

Table 14-7 **Command Description**

Core Command	Description
ut	Generates User Tracking data in XML format.
l2topology	Generates Layer 2 topology data in XML format
discrepancy	Generates discrepancy data in XML format

You must invoke the `cmexport` command with one of the core commands specified in the above table. If no core command is specified, `cmexport` can execute the `-v` or `-h` options only:

- Option `-v` displays the version of the `cmexport` utility.
- Option `-h` (or null option) lists the usage information of this utility.

Arguments and Options

This sections contains:

- [Mandatory Arguments, page 14-19](#)
- [Function-Specific Options, page 14-20](#)

Mandatory Arguments

The options that must be specified with all functions are:

`-u userid`: Specifies the CiscoWorks userid.

Optional Arguments

The options you can specify with any function are:

- `-p password`

Specifies the password for CiscoWorks userid.

If you want to avoid `-p` option which will reveal the password in clear text in CLI, you must store your userid and password in a file and set a variable `CMEXPORTFILE` which points to this file.

You must maintain this file and control access permissions to prevent unauthorized access. `cmexport` looks for current working directory if `CMEXPORTFILE` is set only to the file name instead of to the full path.

If you use the `-p` option, even after setting the `CMEXPORTFILE` variable, the password is taken from the command line instead of from `CMEXPORTFILE`. This is not secure and we recommend that you do not use this option.

The password must be provided in the file in the following format:

userid password

where *userid* is the CiscoWorks user name given in the command line. The delimiter between the *userid* and password is single blank space.

You must provide the delimiter if the password is blank. Otherwise, `cmexport` will not validate the password. The password file can contain multiple entries with different user names. The password that matches the first user name is considered in case of duplicate entries.



Note If `-p password` is used, the password is read from the command line instead of `CMEXPORTFILE`. This is not secure and we recommend that you do not use this option.

- `-d debuglevel`
Sets the debug level based on which debug information is printed. There are two levels of debugging—TRACE and DEBUG. If you do not specify the `-d` option, no logging will occur.
- `-l logfile`
Logs the results of the `cmexport` command to the specified log file name. By default the command output will be displayed in the standard output.

Function-Specific Options

The following function-specific option is supported

`-f filename`

If used with the:

- User Tracking function—Specifies the name of the file to which the user tracking information is to be exported.
- Topology function—Specifies the name of the file to which the layer 2 topology information is to be exported.
- Discrepancy function—Specifies the name of the file to which the discrepancy information is to be exported.

Accessing Help

Enter the following at a CLI:

- `cmexport -h`: Displays a list of options for `cmexport`.
- `cmexport command -h`: Displays a list of options for the `cmexport` command.

On Solaris, you can also enter the following at a CLI:

```
man cmexport
```

DEE Developer's Reference

The following are the schemas used for exporting the user tracking data in XML format:

- [Schema for User Tracking Data, page 14-22](#)
- [User Tracking Schema for Switch Data, page 14-24](#)
- [User Tracking Schema for Phone Data, page 14-26](#)
- [User Tracking Schema for Subnet Data, page 14-27](#)
- [Schema for Topology Data, page 14-29](#)
- [Schema for Discrepancy Data, page 14-31](#)
- [Using Servlet to Export Data from Campus Manager, page 14-32](#)

Schema for User Tracking Data

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2003, 2004 Cisco Systems Inc. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="UTDetails">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CMServer" type="xs:string"/>
        <xs:element name="CreatedAt" type="xs:string"/>
        <xs:element name="SchemaVersion" type="xs:string"/>
        <xs:element name="Heading" type="xs:string"/>
        <xs:element name="Query" type="xs:string"/>
        <xs:element name="Layout" type="xs:string"/>
        <xs:element ref="UTData" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="UTData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UserName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="MACAddress" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="HostName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="IPAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Subnet" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="IPv6Address" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="PrefixLength" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="Prefix" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="DeviceName" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="Device" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Port" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PortName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PortState" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PortDuplex" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="PortSpeed" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="VTPDomain" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="VLAN" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="VLANId" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="VLANType" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="trBRFVLAN" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

        <xs:element name="SecondaryVlan" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="Ring" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Bridge" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="LastSeen" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Notes" type="xs:string" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

User Tracking Schema for Switch Data

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2003, 2004 Cisco Systems Inc. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="UTDetails">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CMServer" type="xs:string"/>
        <xs:element name="CreatedAt" type="xs:string"/>
        <xs:element name="SchemaVersion" type="xs:string"/>
        <xs:element name="Heading" type="xs:string"/>
        <xs:element name="Query" type="xs:string"/>
        <xs:element name="Layout" type="xs:string"/>
        <xs:element ref="SwitchUTData" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SwitchUTData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DeviceName" type="xs:string"/>
        <xs:element name="DeviceIP" type="xs:string"/>
        <xs:element name="UTData" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="UTData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UserName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="MACAddress" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="HostName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="IPAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Subnet" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="IPv6Address" type="xs:string"
minOccurs="0" maxOccurs="1"/>
        <xs:element name="PrefixLength" type="xs:string"
minOccurs="0" maxOccurs="1"/>
        <xs:element name="Prefix" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="Port" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PortName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PortState" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

    <xs:element name="PortDuplex" type="xs:string" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="PortSpeed" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="VTPDomain" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="VLAN" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="VLANId" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="VLANType" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="trBRFVLAN" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="SecondaryVlan" type="xs:string" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="Ring" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Bridge" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="LastSeen" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Notes" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

User Tracking Schema for Phone Data

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="UTDetails">
    <xs:annotation>
      <xs:documentation>It gives the Phone details</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CMServer" type="xs:string"/>
        <xs:element name="CreatedAt" type="xs:string"/>
        <xs:element name="SchemaVersion" type="xs:string"/>
        <xs:element name="Heading" type="xs:string"/>
        <xs:element name="PhoneQuery" type="xs:string"/>
        <xs:element name="PhoneLayout" type="xs:string"/>
        <xs:element ref="PhoneData" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="PhoneData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PhoneNumber" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="MACAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="IPAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="CCMAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Status" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PhoneType" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PhoneDescr" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="DeviceName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Device" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Port" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PortName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="LastSeen" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

User Tracking Schema for Subnet Data

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2003, 2004 Cisco Systems Inc. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="UTDetails">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CMServer" type="xs:string"/>
        <xs:element name="CreatedAt" type="xs:string"/>
        <xs:element name="SchemaVersion" type="xs:string"/>
        <xs:element name="Heading" type="xs:string"/>
        <xs:element name="Query" type="xs:string"/>
        <xs:element name="Layout" type="xs:string"/>
        <xs:element ref="SubnetUTData" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SubnetUTData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SubnetId" type="xs:string"/>
        <xs:element name="UTData" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="UTData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UserName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="MACAddress" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="HostName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="IPAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="IPv6Address" type="xs:string"
minOccurs="0" maxOccurs="1"/>
        <xs:element name="PrefixLength" type="xs:string"
minOccurs="0" maxOccurs="1"/>
        <xs:element name="Prefix" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="DeviceName" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="Device" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Port" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PortName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="PortState" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

```

        <xs:element name="PortDuplex" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="PortSpeed" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="VTPDomain" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="VLAN" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="VLANId" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="VLANType" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="trBRFVLAN" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="SecondaryVlan" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="Ring" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Bridge" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="LastSeen" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Notes" type="xs:string" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

Schema for Topology Data

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="CMData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CMServer" type="xs:string"/>
        <xs:element name="CreatedAt" type="xs:string"/>
        <xs:element name="SchemaVersion" type="xs:string"/>
        <xs:element name="Heading" type="xs:string"/>
        <xs:element ref="Layer2Details" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Layer2Details">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Device" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Device">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DeviceName" type="xs:string"/>
        <xs:element name="IPAddress" type="xs:string"/>
        <xs:element name="DeviceState">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:pattern value="Reachable"/>
              <xs:pattern value="UnReachable"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="DeviceType" type="xs:string"/>
        <xs:element ref="Neighbors" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Neighbors">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Neighbor" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```
</xs:element>
<xs:element name="Neighbor">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="NeighborIPAddress" type="xs:string"/>
      <xs:element name="NeighborDeviceType" type="xs:string"/>
      <xs:element name="Link" type="xs:string"/>
      <xs:element name="LocalPort" type="xs:string"/>
      <xs:element name="RemotePort" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Schema for Discrepancy Data

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="CMData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CMServer" type="xs:string"/>
        <xs:element name="CreatedAt" type="xs:string"/>
        <xs:element name="SchemaVersion" type="xs:string"/>
        <xs:element name="Heading" type="xs:string"/>
        <xs:element ref="Discrepancies" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Discrepancies">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Discrepancy" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Discrepancy">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Category" >
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:pattern value="Physical"/>
              <xs:pattern value="Logical"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="Details" type="xs:string"/>
        <xs:element name="Type" type="xs:string"/>
        <xs:element name="Severity" >
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:pattern value="High"/>
              <xs:pattern value="Medium"/>
              <xs:pattern value="Low"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="Description" type="xs:string"/>
        <xs:element name="FirstFound" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

Using Servlet to Export Data from Campus Manager

The servlet allows you to access DEE features using simple scripts. You can invoke DEE functions by running the script that connects to Campus Manager server and retrieves the data.

You can send the commands to export user tracking, topology, and discrepancy data (`cmexport` and `utexport`) as HTTP or HTTPS requests to a special Campus server URL. This URL identifies a servlet that accepts the request and authenticates the requesting user's identity and credentials before authorizing the information exchange.

- To export User Tracking data, use `UTExportServlet`.
- To export Discrepancy and Layer 2 Topology data, use `CMExportServlet`.
- To invoke `cmexport` and `utexport` commands, the servlet requires a payload file that contains details such as:
 - User credentials
 - The command you want to execute.
 - Optional details such as log and debug options as inputs in XML format.

The servlet then parses the payload file encoded in XML, performs the operations, and returns the results in XML format. You must create the payload file to include the input details and submit it when you ask for servlet access.

Typically, servlet access is used when you need to use the data export feature from a client system.

To use DEE export features, you can write a script to upload the payload file and perform the data export functions. See the following sample scripts:

- [Sample Perl Script to Access the Servlet](#)
- [Sample Java Code to Access the Servlet](#)

For example, if you are using the `script test.pl`, you can invoke the servlet in either of these modes:

- [HTTP Mode](#)
- [HTTPS Mode](#)

HTTP Mode

- For Discrepancy and Layer 2 topology data export, enter:

```
perl test.pl  
http://campus-server:1741/campus/servlet/CMExportServlet  
payload.xml
```

- For User Tracking data export, enter:

```
perl test.pl http://campus-server:1741/cmapps/UTExportServlet  
payload.xml
```

HTTPS Mode

- For Discrepancy and Layer 2 topology data export, enter:

```
perl test.pl https://campus-server/campus/servlet/CMExportServlet  
payload.xml
```

- For User Tracking data export, enter:

```
perl test.pl https://campus-server/cmapps/UTExportServlet  
payload.xml
```

Sample Perl Script to Access the Servlet

```
#!/opt/CSCOpX/bin/perl

use LWP::UserAgent;
$| = 1;
$temp = $ARGV[0] ;
$fname = $ARGV[1] ;
if ( -f $fname ) {
open (FILE,"$fname") || die "File open Failed $!";
while ( <FILE> )
{
    $str .= $_ ;
}
close(FILE);
}
url_call($temp);

#-- Activate a CGI:
sub url_call {
    my ($url) = @_ ;
    my $ua = new LWP::UserAgent;
    $ua->timeout(5000);
    my $hdr = new HTTP::Headers 'Content-Type' => 'text/html';
    my $req = new HTTP::Request ('GET', $url, $hdr);
    $req->content($str);
    my $res = $ua->request($req);
    my $result;
    if ($res->is_error)
    {
        {
            print "ERROR : ", $res->code, " : ", $res->message, "\n";
            $result = '';
        }
    }
    else
    {
        {
            $result = $res->content;
            if($result =~ /Authorization error/)
            {
                print "Authorization error\n";
            }
            else
            {
                print $result ;
            }
        }
    }
}
}
```

Sample Java Code to Access the Servlet

```
import java.io.*;
import java.net.URL;
import java.net.HttpURLConnection;
import java.lang.String;
import java.lang.Byte;

class CMExportServletRun {

    static void main (String args[])
    {
        try {
            URL url = new
URL("http://localhost:1741/campus/servlet/CMExportServlet");

            String payload = "adminadminut_hostdee.log1";

            HttpURLConnection con;
            InputStream is;
            //opens connection to servlet
            con = (HttpURLConnection)url.openConnection();
            con.setRequestMethod("POST");
            con.setRequestProperty("Content-type", "text/xml");
            con.setDoOutput(true);
            con.setUseCaches(false);

            OutputStream bos = new
BufferedOutputStream(con.getOutputStream());
            PrintWriter out = new PrintWriter(bos);
            out.println(payload);
            out.flush();
            out.close();

            //prints out response from CMExportServlet
            byte [] strBytes=new byte[10];
            int noOfBytes = 0;

            is = con.getInputStream();

            BufferedReader bfr = new BufferedReader(new
InputStreamReader(is));
            String str = null ;

            while ( ( str = bfr.readLine() ) != null ) {
                System.out.println(str);
            }
        }
        catch (Exception e) {
```

```

        System.out.println(e.toString());
    }
}
}

```

Payload File

The payload file is an XML file, which contains inputs required for the DEE servlet to process requests for data export. Schema for the payload XML file is given in Schema for Payload File.

[Table 14-8](#) describes the elements in the schema.

Table 14-8 *Elements in the Schema*

Element	Description
username	CiscoWorks user name.
password	Password for CiscoWorks username.
command	Command inside this tag can be ut_host, ut_phone, l2topology or discrepancy.
view	(Optional) You can use this option when you specify ut_host. This specifies the presentation of the User Tracking data in the hierarchical format with either switch or subnet as the root.
queryname	User Tracking host data is exported in XML format for the query provided in queryname. You can use this option when you specify ut_host
layoutname	User Tracking host data is exported in XML format for the layout provided in layoutname. You can use this option when you specify ut_host
queryphone	User Tracking phone data is exported in XML format for the query given in queryphone. You can use this option when you specify ut_phone

Table 14-8 *Elements in the Schema (continued)*

Element	Description
layoutphone	User Tracking phone data is exported in XML format for the layout given in layoutPhone. You can use this option when you specify ut_phone
debug	Optional. Debug messages can be collected only if log file is specified in the log option. The debug level could be 1 or 2. You can set the value to: 1—For basic debug information. 2—For detailed debug information.

This section also describes:

- [Sample Payload File, page 14-37](#)
- [Schema for Payload File, page 14-38](#)

Sample Payload File

```
<payload>
  <username>username</username>
  <password>password</password>
  <command>ut_host</command>
  <debug>1</debug>
  <view></view>
</payload>
```

Schema for Payload File

You can use the following schema for creating the payload file in XML format.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="payload">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="username" type="xs:string"/>
        <xs:element name="password" type="xs:string"/>
        <xs:element name="command" type="xs:string"/>
        <xs:element name="view" type="xs:string"/>
        <xs:element name="queryname" type="xs:string"/>
        <xs:element name="layoutname" type="xs:string"/>
        <xs:element name="queryphone" type="xs:string"/>
        <xs:element name="layoutphone" type="xs:string"/>
        <xs:element name="debug" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Frequently Asked Questions

Following is a list of frequently asked questions about Campus Manager DEE:

- [Where does DEE collect the Discrepancy information from?, page 14-39](#)
- [What is an XSD file?, page 14-39](#)
- [How can I make use of the servlet interface?, page 14-40](#)
- [How can I get user tracking and Layer 2 topology data for a particular set of switches or subnets managed by separate Campus Manager servers?, page 14-40](#)
- [I do not want to provide the password in the command line as it is insecure. Is there a way to provide it in a secure way?, page 14-40](#)
- [Where will the XML output file be stored?, page 14-40](#)
- [Why am I getting a parse error when trying to parse some of the output files?, page 14-41](#)

Q. Where does DEE collect the Discrepancy information from?

A. DEE collects the running Discrepancy data from the latest configuration in Campus Manager server.

Q. What is an XSD file?

A. XSD file is an XML based alternative to Document Type Definition (DTD). It is based on XML schema which describes the structure of an XML document. An XML schema defines the valid building blocks of an XML document, similar to a DTD.

An XML Schema:

- Defines elements that can appear in a document.
- Defines attributes that can appear in a document.
- Defines which elements are child elements.
- Defines the order of child elements.
- Defines the number of child elements.

- Defines whether an element is empty or can include text.
- Defines data types for elements and attributes.
- Defines default and fixed values for elements and attributes.

For more information, see W3Schools Online Web Tutorials site.

- Q. How can I make use of the servlet interface?
- A. You must write customized scripts that can connect to the servlet. The arguments and options have to be specified in XML format. Details can be found in [“Using Servlet to Export Data from Campus Manager” section on page 14-32](#).
- Q. How can I get user tracking and Layer 2 topology data for a particular set of switches or subnets managed by separate Campus Manager servers?
- A. No. This feature is not supported.
- Q. I do not want to provide the password in the command line as it is insecure. Is there a way to provide it in a secure way?
- A. Yes. You can create an environment variable *CMEXPORTFILE*, which points to a text file that contains the userid and password list separated by a blank space.
- Q. Where will the XML output file be stored?
- A. The default locations for storing the XML output files are:
- PX_DATADIR/cmexport/ut/timestamp.xml
 - PX_DATADIR/cmexport/l2topology/timestampL2Topology.xml
 - PX_DATADIR/cmexport/Discrepancy/timestampDiscrepancy.xml
- You can use the **-f** option to specify an alternative location.

- Q. Why am I getting a parse error when trying to parse some of the output files?
- A. A few classes in Optical switches contain special characters with ASCII code higher than 160. Most of the XML parsers do not support these characters and hence fail to parse them.

To overcome this, you have to manually search for those elements with special characters and append CDATA as given in the example below:

If there is an element

```
<checksum> φŪo </checksum>
```

Change it to:

```
<checksum> <![CDATA [φŪo ] ]> </checksum>
```

