Oracle® COREid Access and Identity

Customization Guide

10*g* Release 2 (10.1.2)
Part No. B19012-01

May 2005

ORACLE

This program contains third-party code from Apache. Under the terms of the Apache Software License, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Apache software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Apache software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Apache.

Contents

Preface

This *Customization Guide* explains how to control the way COREid operates, by making configuration changes to operating systems or Web or directory servers, editing the content of certain XML files, or changing directory content. It also provides an overview, from an Administrator's point of view, of the Access Server API and the Authorization and Authentication Plug-in APIs.

Note: Oracle *COREid* was previously known as Oblix *Netpoint*. All legacy references to Oblix and NetPoint, for example, in screen shots, illustrations, and documentation titles, should be understood to refer to Oracle and COREid, respectively.

This Preface covers the following topics:

Intended Audience

This guide is intended for anyone who needs to customize COREid. Topics here assume that you have some prior experience using Oracle products, understand the logical connections between Identity and Access components, and have a general knowledge of directories and LDAP. You must also be comfortable manipulating files and running applications at the command line level. Techniques provided here are vulnerable to error and should be used with the utmost care.

COREid Documentation

The manuals that are available for this release include:

Introduction to COREid—Provides an introduction to COREid, a road map to COREid manuals, and a COREid glossary of terms.

COREid Release Notes—Provides up-to-the minute details about the latest COREid release.

COREid Installation Guide—Explains how to install and configure the COREid components.

COREid Upgrade Guide—Explains how to upgrade earlier versions of COREid to the latest version of COREid.

COREid Administration Guide—Explains how to configure COREid applications to display information stored in the directory, how to assign view and modify permissions for data displayed on the COREid applications, and how to assign access controls to users.

COREid Deployment Guide—Provides information for people who plan and manage the environment in which COREid runs. This guide covers capacity planning, system tuning, failover, load balancing, caching, and migration planning.

COREid Customization Guide—Explains how to change the appearance of COREid applications and how to control COREid by making changes to operating systems, Web servers, directory servers, directory content, or by connecting CGI files or JavaScripts to COREid screens. This guide also describes the Access Server API and the Authorization and Authentication Plug-in APIs.

COREid Developer Guide—Explains how to create AccessGates and how to develop plug-ins. This guide also provides information to be aware of when creating CGI files or JavaScripts for COREid.

COREid Integration Guide—Explains how to set up COREid to run with third-party products such as BEA WebLogic, the Plumtree portal, and IBM Websphere.

COREid Schema Description—Provides details about the COREid schema.

Online Help is available from each COREid screen.

Typographical Conventions

COREid manuals use the following typographical conventions:

- When you are instructed to select elements sequentially, the actions are separated with angle brackets, as shown below:

  Click System Admin > System Configuration > View Server Settings.

- Paths to a file are shown using syntax for either the Unix or Windows platform:

  /COREid_install_dir/identity/oblix/logs/debugfile.lst
  COREid_install_dir\identity\oblix\logs\debugfile.lst

  where *COREid_install_dir* refers to the directory where the component, in this case, the COREid Server, is installed.

Contact Information

For a list of contacts including corporate offices world wide, sales, and other details, visit the Oracle Web site at:

`http://www.oracle.com`

You can contact Oracle with questions or comments as follows:

Customer Care—http://www.oracle.com/support/contact.html

Corporate Headquarters

Oracle maintains offices world wide. Oracle corporate headquarters is located at:

500 Oracle Parkway
Redwood Shores, CA 94065
Phone: (650) 506-7000

Before Contacting Customer Care

Before contacting Customer Care, please have available the following:

- Oracle product name and version number
- Type of computer and operating system you are using

Accessing the Customer Care Knowledge Base

For more information about using COREid, see the Oracle Customer Care Knowledge Base. To access the Knowledge Base, you need a login name and password, which you can obtain from your Oracle sales representative.

To access the Knowledge Base:

1. Enter the following URL in your browser and press Return.

   ```
   http://www.oracle.com/support/contact.html
   ```

2. Click the phrase, Login to the Oracle PremiumCare Online Portal.
3. Enter your user name and password in the box that appears, then click Login.
4. Under Oracle Support Tools, click Case Manager.
5. In the next screen, click Find Answers to gain access to the Knowledge Base.

1 Introduction

This book explains how to control the way NetPoint operates, by making configuration changes to operating systems or Web or directory servers, editing the content of certain XML files, or changing directory content. It also provides an overview, from an Administrator's point of view, of the Access Server API and the Authorization and Authentication Plug-in APIs. Topics include:

- Changing the appearance of NetPoint applications
- Designing the GUI by editing XML files
- Modifying catalog files
- Connecting CGI files or JavaScripts to NetPoint screens
- Controlling how NetPoint operates by making configuration changes to the operating system, Web or directory servers, or directory content
- Introducing Access Server API and the Authorization and Authentication Plug-in APIs from an Administrator's point of view

Techniques here are vulnerable to error and should be used with the utmost care. This guide assumes that you have some prior knowledge of and experience with:

- Using NetPoint
- Logical connections between the COREid and Access systems
- General working knowledge of directories and LDAP
- Comfort manipulating files and running applications at the command-line level

Other helpful experience includes:

- System and/or database administration
- Familiarity with CGI files or JavaScripts
- Familiarity with your Web server, Web browser, operating system, and configuration details

Before you begin using the information in this book, NetPoint should be installed and its operation confirmed. For details, see the NetPoint 7.0 Installation Guide.

2 Designing the GUI with PresentationXML

The COREid System combines XSL (eXtensible Style Language) stylesheets and XML (eXtensible Markup Language) data to dynamically create almost all of the pages presented to its users. This capability, which Oblix calls *PresentationXML*, provides NetPoint developers with a great deal of design flexibility and avoids the necessity of providing many pages of static HTML content along with the product.

Within the bounds described in this chapter, you can use this feature yourself to customize COREid System user application presentation to suit your own needs. For example, you can:

- Apply your organization's color schemes and other graphical style elements such as fonts, button images, and logos, to NetPoint pages.
- Add, modify, or remove particular functions on a COREid page.
- Add hidden information which could be used by the Identity Event Plug-in API (see the *NetPoint 7.0 Developer Guide*).
- Create entirely new pages and functionality.

This chapter tells you how to use PresentationXML.

- See "PresentationXML Operation" on page 16 to understand how PresentationXML works.
- See "PresentationXML Components" on page 25 for a description of the essential parts of PresentationXML.
- See "PresentationXML Libraries" on page 39 for a listing and description of some major parts of the full library of PresentationXML components.
- See "Customizing NetPoint" on page 80 for an outline of one method for doing customization.
- See "Customizing the NetPoint GUI" on page 85 for an example you can complete to gain first-hand experience with customizing NetPoint.

Note: Prior experience with XSL and XML is not essential to using PresentationXML. However, you will need to learn and understand them as you delve into more complex kinds of changes. Some reference sites and brief syntax introduction are provided in “XML Background” on page 171.

The System Admin Console always uses the NetPoint default, Classic Style.

PresentationXML Operation

This section describes how standard PresentationXML operates, introduces some parameters that can be used to control its output, and describes a useful alternate mode of operation. The terms you will see include:

Stylesheet—This term identifies XSL stylesheets that describe how data sent over the Web is to be presented to the user.

Base Stylesheet—This term refers to several NetPoint stylesheets that provide a foundation for other stylesheets:

basic.xsl
font.xsl
searchform.xsl
navbar.xsl
title.xsl

Wrapper—This term identifies an XSL stylesheet file that contains only XSL include statements with pointers to other files. For example:

For more information, see “PresentationXML Components” on page 25 and “Styles” on page 35.

Server-Side Processing

The diagram on the next page shows the default data flow for PresentationXML. This default process, called *server-side processing*, generates HTML and presents it to the browser. The following steps are performed.

Process overview: Server-side processing

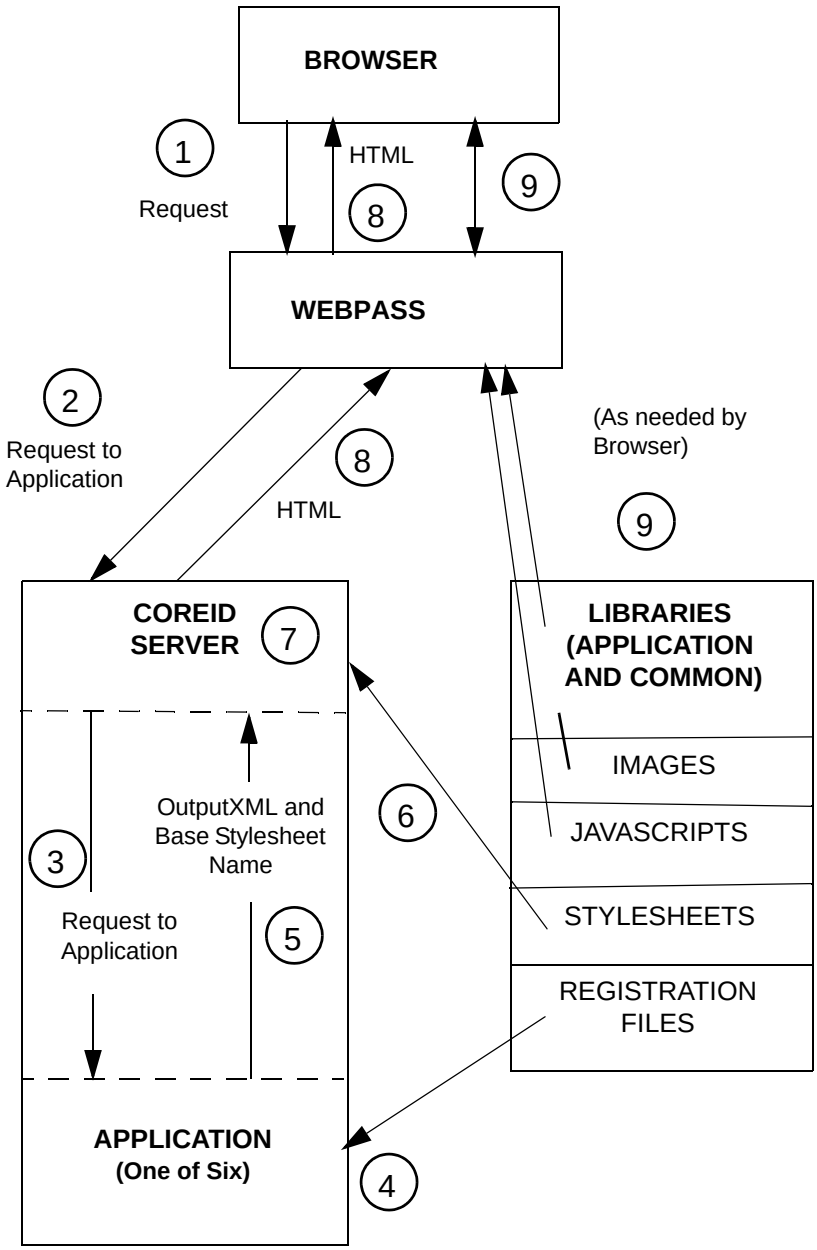
1. The browser sends a request to the URL of a Web server that includes the WebPass plug-in.

   The full URL contains the location of the Web server and implicitly the application, such as the Group Manager, that is expected to service the request. The URL also usually includes information telling the application what task to perform and providing parameters that direct the task. (See more on the full syntax for the URL on page 26.)

2. WebPass takes the request, makes a few minor changes to it, and passes it to the COREid Server.

3. The COREid Server passes the request to the appropriate application. (For the sake of clarity, the diagram shows the application as if it were a separate entity, but it is actually a dynamically loaded part of the COREid Server.)

4. The application processes the request and creates an XML file, called OutputXML, which contains information that will appear as part of the final HTML. The application also opens its *registration file* (page 31) to get the name of the base XSL stylesheet that applies to the request that it has just satisfied.



5. The OutputXML and the name of the base stylesheet are returned to the COREid Server.

6. The COREid Server reads the text of the base stylesheet from the library. This text usually includes references to other stylesheets, which are used in common by many different types of requests. For each reference, the COREid Server

reads the additional text from the stylesheet library and inserts it in-line in place of the reference, making one large stylesheet.

7. An XSLT (XSL transformer) application is part of the COREid Server. The XSLT parses and interprets the stylesheet and combines it with the OutputXML to create HTML. The OutputXML content provides a basis for the decisions that are made as the stylesheet is interpreted and also provides data to be included in the HTML.
8. When the entire stylesheet has been processed, the resulting HTML is sent to WebPass, which returns it to the browser.
9. The browser uses WebPass to obtain GIF images and JavaScripts as needed.

Parameters that Control Operation

You can append any of three parameters, `format`, `xsl`, and `style`, to the URL to shape the way the XSLT works with the OutPutXML and the stylesheets.

Format Parameter

The `format` parameter can be used to control the way in which the COREid Server combines the OutputXML and the stylesheet before the resulting information is passed to the browser, *if* this is allowed by the setting of a certain parameter in the `globalparams.xml` parameter file. (See page 211.) The COREid Server checks the value that was provided for the `outputFormat` parameter in this file. The parameter must be set to `default` in the parameter file to enable use of the `format` parameter here in the PresentationXML URL.

The format parameter takes one of three values:

- **Default**—If the parameter is not included in the URL XSLT processing is done at the server.
- **format=xmlnoxsl**—The COREid Server returns the Output XML without doing XSLT processing; that is, the XSL stylesheet is not applied. This is a good way to generate the OutputXML. If you do this, and want to capture the result, save the displayed data *as XML* (if your browser supports this). The true OutputXML contains escaped characters that will be lost if you save the displayed data as a text file.

Note: If the same CGI is used to handle different functionality, just appending "&format=xmlnoxsl" to the URL would result in executing the default functionality of a given CGI.

If the same CGI is used to handle different functionality, you need a way to re-submit the form request with "&format=xmlnoxsl" as part of the action. This can be achieved using the methods described below.

- **Method 1**—Append “&format=xmlnoxsl” to the form action within the relevant .xsl stylesheets. While this may seem straight-forward enough, be aware that you are making changes to a stylesheet and this could have unintended consequences. After making the change, you need to either restart the COREid Server or update the globalparams.xml file to disable stylesheet caching (for example, set stylesheet caching to 1) and allow stylesheet dynamic-change updates.
- **Method 2**—Rewrite the POST request as a GET request, and add &format=xmlnoxsl. It is not always possible to tell directly what paramaters are being passed to the POST request.

  Sometimes the action is a JavaScript that rewrites the form parameters before the form request gets submitted. In such instances, you may want to run a packet sniffer to capture the POST request as it is leaving the browser. Alternatively, you can enable debugging on the COREid Server and look at the request data. However, you may have to sift through a certain amount of non-relevant data. You then recreate the POST request as a GET request because it is easier to submit as one URL. In some rare cases, the POST data is too long and does not fit as a GET request, so you can write an HTML static form and manually fill-in the data from the POST request captured with the sniffer or from the COREid debug log.

  Note: Oblix has found certain packet sniffers useful. For example, daSniff available at http://demosten.com/dasniff offers a command-line interface and requires the WinPcap libraryavailable at http://netgroup-serv.polito.it/winpcap.

- **format=xml**—The COREid Server returns the Output XML, with the name of the base XSL stylesheet embedded as an XML element. For this to be useful, the browser must be able to do its own XSLT processing. In that case the browser sends requests to WebPass for the content of included stylesheets, and WebPass gets them directly from the appropriate library.

The only browser currently able to do its own XSLT processing is Microsoft’s Internet Explorer (IE). IE Versions 5.5 and later are compatible with PresentationXML.

Earlier versions of IE use an earlier proposed version of XSLT, based on a working draft from 1998. This is inconsistent with the current version of XSLT, which COREid follows. To use the `format=xml` option with the earlier versions of IE, you *must* either rewrite the XSL stylesheet to be consistent with the earlier draft version or download Microsoft’s patch to its XSLT processor. See “XML Background” on page 171 for a discussion of how to download and install the patch.

XSL Parameter

The xsl parameter determines which XSL stylesheet is to be combined with the OutPutXML. It can take either of two values:

- **Default**—If the parameter is not included in the URL, the default is to apply the XSL stylesheet specified in the registration file.
- **xsl=stylesheet_name**—Use the specified stylesheet as the base stylesheet, in place of the one specified in the Registration File.

Style Parameter

The style parameter indicates which style directory in the library to get the base stylesheet from (For an explanation of styles, see “Styles” on page 35.). Once the style parameter has been used in the URL, the style you chose is implicitly included in all further requests in the session for that particular browser.

- **Default**—If the parameter is not included in the URL, the default is to either: get the base stylesheet from the style0 directory (if the style parameter has not been previously used), or continue to get the base stylesheet from the style directory specified by a previous use of the style parameter).
- **style=styledirectoryname**—Get the base stylesheet from the specified style directory, and continue to do so for the rest of the session with this browser.

Note: Administrators have the ability to create new style directories and set them as the default style. In the above discussion, style0 stands for the default style directory.

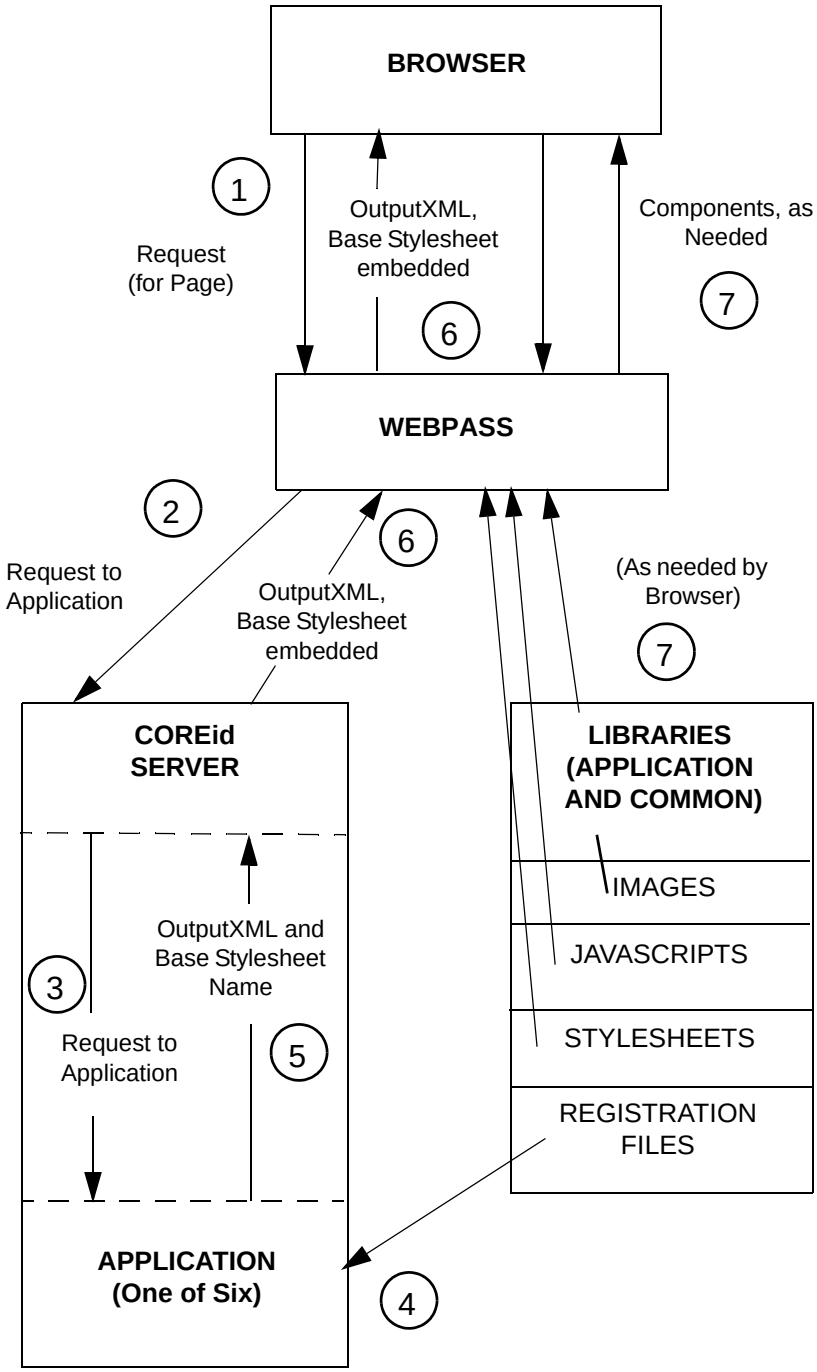
Client-Side Processing

The diagram on the following page shows the major alternative method for using PresentationXML. This method is called *client-side processing*, because it presumes that the browser will generate its own HTML, given the OutPutXML and the base stylesheet. To set this up, you use the format=xml parameter as discussed earlier. In this case, the only responsibility of the COREid Server is to pass the OutPutXML, with the name of the base stylesheet embedded, through WebPass back to the browser. The browser then sends requests to WebPass for various library components as it needs them.

Process overview: Client-side processing

1. The browser sends a request to the Web server, as in server-side processing.
2. WebPass takes the request, makes a few minor changes to it, and passes it to the COREid Server, as before.

3. The COREid Server passes the request to the appropriate application, as before.
4. The application processes the request, creates an OutPutXML file, and retrieves the base stylesheet, as before.
5. The OutputXML and base stylesheet are returned to the COREid Server, as before.
6. However, the COREid Server does no processing of the OutPutXML. Instead, it embeds the base stylesheet name into the OutPutXML and sends the result to Webpass, which passes it the browser.
7. The browser then makes requests to the Web server, as needed, for the referenced stylesheets, images and JavaScripts.


BROWSER
1
Request
(for Page)
OutputXML,
Base Stylesheet
embedded
6
Components, as
Needed
7
WEBPASS
2
Request to
Application
6
OutputXML,
Base Stylesheet
embedded
(As needed by
Browser)
7
COREid
SERVER
LIBRARIES
(APPLICATION
AND COMMON)
IMAGES
JAVASCRIPTS
STYLESHEETS
REGISTRATION
FILES
OutputXML and
Base Stylesheet
Name
3
Request to
Application
5
APPLICATION
(One of Six)
4

Caching Considerations

For the sake of efficiency, the COREid System maintains a cache of stylesheets, converted to a binary format. Stylesheets can be changed while the COREid System is running, but will need to be forced into this cache in order to take effect. One way to do this is to stop and start the COREid Server.

Another way is to use the XSL stylesheet control parameters provided (page 211) in the globalparams.xml parameter file:

COREid_install_dir\identity\oblix\apps\common\bin\globalparams.xml

where *COREid_install_dir* is the directory where COREid is installed.

There are two of these:

- **XSLStylesheetLiveUpdate**—The default value for this parameter is false, which means, if the stylesheet is cached, the COREid System uses the cached binary version regardless of any changes to the original file. In a production environment, this parameter should be set to false. Setting it to true causes NetPoint to compare filestamps to see if the stylesheet has been modified in between requests. This is not desirable in a production system because it is an unnecessary overhead.

  Changing the value to true causes a check of the timestamp of the text version of the file against the timestamp for the cached version. If the cached version is older, the text file is converted to binary and replaces the older version in the cache. Note, however, that this works only for base stylesheets.

- **XSLStylesheetCacheSize**—Default value for this parameter is 200. Change the value to 1, so that only one stylesheet can be cached (0 is not a legal value). Go forward one page, then return to the one you are testing. This works for stylesheets at all levels.

  Recommendation: The XSLStylesheetCacheSize parameter should be at least equal to or greater than the number of base stylesheets. Ideally the default value, which is 200 for this parameter, should be good enough for most cases.

PresentationXML Components

This section describes the COREid components that work together to create PresentationXML. These components are:

XSL Transformer—In addition to transferring data between WebPass and the applications, the COREid Server is usually responsible for transforming the OutPutXML (using the stylesheet as a guide) into HTML for use by the browser. To accomplish this, the COREid Server contains a built-in XSL Transformer. See “XSL Transformer” on page 25.

Applications—The functional entities within the COREid Server, such as the User Manager or Lost Password Management, that handle the logic specific to each request. See “NetPoint Applications” on page 26.

URLs—The location information, and other parameters, that the browser provides in order to make a request through WebPass to reach a specific application. See “URLs” on page 26.

Output XML—The stream of XML data created by each COREid application. This can be captured as a file. It contains the information to be shown on the requested page. See “OutPutXML” on page 28.

XML Schemas—Files that describe the type and hierarchy of data that appears in the OutPutXML. See “XML Schemas” on page 28.

Registration files— Files that specify which stylesheets to use and which XML schema file describes the OutPutXML. See “Registration Files” on page 31.

JavaScripts—JavaScript applications which perform specialized tasks. Pointers to these are provided within the HTML. See “JavaScripts” on page 35.

Styles—Collections of data that support the generation of the HTML for the page. For each application, the data includes:

- **Stylesheets**—Files containing XSL instructions which tell the XSL transformer how to use the information provided in the OutPutXML. See “XSL Stylesheet Content” on page 36.
- **Images**—Small graphic files referenced in the HTML, which are combined to make the final GUI presentation. See “Images” on page 38.

XSL Transformer

The COREid Server contains a built in XSL Transformer. The Transformer follows the logic provided in an XSL stylesheet and creates a file of text following an HTML format. The Transformer uses the content of the OutPutXML to determine branches at decision points in its logic and to get content for data to be included in the HTML.

NetPoint Applications

PresentationXML supports six applications, which are dynamically loaded and then executed from within the COREid Server. The following table lists the applications and shows the `appname` for each application. The browser provides a URL as part of its request to the COREid Server and the `appname` appears as a string within the URL. The `appname` also appears as a directory name, under which the PresentationXML data for the particular application can be found (see “Directory Structure” on page 39). Extensive descriptions of each application and details for using each one are provided in the *Introduction to NetPoint 7.0 Guide*.

Application	appname
Group Manager	groupservcenter
Organization Manager	objservcenter
User Manager	userservcenter
Lost Password Management	lost_pwd_mgmt
Query Builder	querybuilder
Selector	selector

There is an additional legal value which is used in URLs for activities that are common across applications:

Application	appname
Resources used across applications	common

URLs

URLs that the browser provides to the COREid Server are of two basic kinds. First is the URL for the first page of each application, the one you arrive at after logging in. An example is the front page URL for User Manager:

http://www.customer.com/identity/oblix/apps/userservcenter/bin/userservcenter.cgi

This can be divided into several parts. The first part is:

http://www.customer.com/

which is the location of the server site to which the COREid Server has been added as a plug-in.The second part:

http://www.customer.com/identity/oblix/

tells the server site that you want to work with the COREID Server. The third part is:

apps/userservcenter/bin/userservcenter.cgi

which tells the COREid server which of its applications, in this case the User Manager, you want to work with.

Formally, per the specification *Internet RFC 2396- Uniform Resource Identifiers (URI): Generic Syntax*, the string below:

identity/oblix/apps/userservcenter/bin/userservcenter.cgi

is a path to an application that can be executed.

The following table shows the path to the first page URL for each application:

Application	Application Front Page URL
Group Manager	identity/oblix/apps/groupservcenter/bin/ groupservcenter.cgi
Organization Manager	identity/oblix/apps/objservcenter/bin/objservcenter.cgi
User Manager	identity/oblix/apps/userservcenter/bin/userservcenter.cgi
Lost Password Management	identity/oblix/apps/lost_pwd_mgmt/bin/lost_pwd_mgmt.cgi
Query Builder	identity/oblix/apps/querybuilder/bin/querybuilder.cgi
Selector	identity/oblix/apps/selector/bin/selector.cgi

The front page in turn contains links to other pages within the application. The URL content for such a link could be something like the following:

```
http://www.customer.com/identity/oblix/apps/
userservcenter/bin/userservcenter.cgi
?program=view
&uid=cn%3DJohn%20Smith%2C%20ou%3DCorporate
     %2C%20ou%3DCompany%2C%20c%3DUS
&tab_id=Employees
```

This an extension of the content of the front page URL. The information locating the server and specifying which application to use are as they were before. Added are several *parameters*, set off by the `?` and `&` delimiters. (Parameters are discussed in more detail at “Customizing Portal Inserts” on page 101.)

For PresentationXML, the most significant parameter is `program`, which identifies the purpose of the requested page. The program name serves as a lookup index in a registration file for each application. Registration files are covered in more detail at “Registration Files” on page 31.

OutPutXML

OutPutXML is a structured stream of information which contains the content of a page to be created by using Presentation XML. The arrangement of this content and the choice as to how much of it will appear on the final page is determined by the stylesheet that is applied to it.

Each application generates an OutPutXML stream matching the task that it is performing. The content of this stream varies significantly with the directory content that the application is working with.

The detailed content of the OutPutXML is therefore not predictable. However, it is possible to predict the *kinds* of data that an OutPutXML stream will contain, because this data conforms to *XML schemas* which NetPoint follows for each application.

Given the application and the program name, you can locate the corresponding schema file name in the application's registration file.

XML Schemas

The OutPutXML information generated by each program within each NetPoint application is hard-coded and is not directly changeable by users. The content of the OutPutXML is *not* controlled by the content of the XML schema file. The file is provided only as a developer's aid, to help you design XSL stylesheets to work with the OutPutXML. XML schema files follow a standard developed by the World Wide Web Consortium. See page 173 for the standard, and an introduction to XML syntax.

Note: XML schema files do *not* have any effect on the PresentationXML and are located only on WebPass. Oblix recommends that you *not* change any of the XML schema files.

Here is an example XML schema file, the complete usc_profile.xsd file, located in *WebPass_install_dir*\identity\oblix\WebServices\XMLSchema\usc_profile.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) by zzz (zzz) -->
<xsd:schema
     targetNamespace="http://www.oblix.com/" xmlns:xsd="http://www.w3.org/2001/
XMLSchema" xmlns="http://www.oblix.com/" elementFormDefault="qualified">
   <xsd:include schemaLocation="component_profile.xsd"/>
</xsd:schema>
```

The line in bold beginning with `xsd:include` indicates that there is more information to be included in-line in this schema definition, in the file at:

../..//XMLSchema/component_profile.xsd.

component_profile.xsd

The complete content of the component_profile.xsd file follows. You will notice that this file includes pointers to four other schema files, such as navbar.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.oblix.com/" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns="http://www.oblix.com/"
elementFormDefault="qualified">

   <xsd:include schemaLocation="navbar.xsd"/>
   <xsd:include schemaLocation="searchform.xsd"/>
   <xsd:include schemaLocation="component_panel.xsd"/>
   <xsd:include schemaLocation="component_basic.xsd"/>
   <xsd:element name="ObProfile">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element ref="ObPanel" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element ref="ObHeaderPanel" minOccurs="0"/>
         <xsd:element ref="ObRequestInfo" minOccurs="0"/>
         <xsd:element ref="ObScripts" minOccurs="0"/>
         <xsd:element ref="ObForm" minOccurs="0"/>
         <xsd:element ref="ObDisplay" minOccurs="0"/>
         <xsd:element ref="ObTextMessage" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element ref="ObSelectorInfoForm" minOccurs="0"/>
         <xsd:element ref="ObButton" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element ref="ObStatus" minOccurs="0"/>
       </xsd:sequence>
     </xsd:complexType>
   </xsd:element>
   <xsd:element name="Oblix">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:choice>

           <xsd:element ref="ObProfile"/>
           <xsd:element ref="ObError"/>
         </xsd:choice>
         <xsd:element ref="ObNavbar" minOccurs="0"/>
         <xsd:element ref="ObSearchForm" minOccurs="0"/>
```

```
          <xsd:element ref="ObApplicationFunc" minOccurs="0"/>
          <xsd:element ref="ObStatus" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="oblang" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

This file is representative of most of the XML schema files in that:

- It begins with a set of included XML schemas, ones that are used in common across several applications.
- The lines starting with <xsd:element name="ObProfile"> and ending with </xsd:element> define an element called ObProfile.
- The lines in between, starting with <xsd:element ref= indicate that the ObProfile element contains the nested elements ObPanel, ObHeaderPanel, and so on.

Note: ObProfile in turn is referenced as part of the Oblix element, which is the root element of the entire Oblix schema. The root element will be the starting point for the application of the XSL stylesheet.

Schema Files

XML schema files do not have any effect on the PresentationXML and are located only on WebPass.

Listed below are some of the most often used schema files:

- **navbar.xsd**—Defines the Navigation Bar, which is the top two lines of each page, including the application name, help and logout buttons, and the various tabs to select other applications or modules within the application.
- **searchform.xsd**—Defines the Search Form line, which is the row of graphics shown on some pages, that starts with the graphic labeled search. This row may have other rows beneath it.
- **component_panel.xsd**—Defines the content for profiles, the sets of descriptive information for users, groups or organizations.
- **component_basic.xsd**—Defines many of the lowest level elements, and includes displaytype.xsd and error.xsd.
- **displaytype.xsd**—Defines formatting for each of the COREid display types.
- **error.xsd**—Defines the `ObError` element.

Each of these files contains other nested elements, and so on. For easy reference, tables giving the names of the most commonly used schema files, the names of the elements contained within them, and a brief description of what each element is, are provided in the “XML Schema Elements Library” on page 57.

Note: The OutPutXML information generated by each program within each NetPoint application is hard-coded and is not directly changeable by users. The content of the OutPutXML is *not* controlled by the content of the XML schema file. The file is provided only as a developer’s aid, to help you design XSL stylesheets to work with the OutPutXML. Users should *not* change any of the XML schema files.

Registration Files

Each of the COREid applications has associated with it a unique registration file. This is a text file holding information arranged as a series of XML elements. Using the ObProgram name that appears in the URL for a lookup index, the specific COREid application searches its registration file to get the name of the base stylesheet that is to be applied to the OutPutXML. As an aid to the developer, the registration file also holds the name of the XML schema that describes the OutPutXML for the `program`.

The following table shows the path to the registration file for each application.

Application	Registration File Location
Group Manager	*COREid_install_dir*\identity\oblix\apps\groupservcenter\bin\groupservcenterreg.xml
Organization Manager	*COREid_install_dir*\identity\oblix\apps\objservcenter\bin\objservcenterreg.xml
User Manager	*COREid_install_dir*\identity\oblix\apps\userservcenter\bin\userservcenterreg.xml
Lost Password Management	*COREid_install_dir*\identity\oblix\apps\lost_pwd_mgmt\bin\lostpwdmgmtreg.xml
Query Builder	*COREid_install_dir*\identity\oblix\apps\querybuilder\bin\querybuilderreg.xml
Selector	*COREid_install_dir*\identity\oblix\apps\selector\bin\selectorreg.xml

The main applications also frequently call logic from the common resource applications. When this happens the application looks for the appropriate stylesheet in the common registration file:

Application	appname
Common resources	*COREid_install_dir*\identity\oblix\common\bin\oblixbasereg.xml

General Content of Registration Files

Registration files have the following general content. In the example below, key variable values within each registration file are indicated in *italic* text. Structural elements in the example below appear in **bold**.

```
<?xml version="1.0"?>
<ObProgramRegistry>
 <ObApplication name="application_name">
  <ObProgram name="a_program_name">
       <ObButton name="a_button_name"/>
       <ObButton name="another_button_name"/>
       <ObButton name="yet_another_button_name"/>
       <ObButton name="maybe_more_button_names"/>
       ...
       <ObStyleSheet name="stylesheetname.xsl"/>
       <ObSchema name="XML_schema_name.xsd"/>
  </ObProgram>
  <ObProgram name="another_program_name">
       <ObButton name="a_button_associated_with_it"/>
       <ObStyleSheet name="Its_stylesheetname.xsl"/>
       <ObSchema name="Its_XML_schema_name.xsd"/>
  </ObProgram>
  <ObProgram name="and_so_on">
       ...
  </ObProgram>
       ...
</ObApplication>
</ObProgramRegistry>
```

Each of these elements serves a particular purpose.

For example, the following element identifies the file as a registration file:

```
<ObProgramRegistry>
```

while the element below identifies the application to which this registration file applies:

```
<ObApplication name="the_application_name">
```

For example:

```
<ObApplication name="groupservcenter">
```

The program name identifies the `program,` or function, within the application, to which the stylesheet, buttons and schema apply:

```
<ObProgram name="a_program_name">
```

For example:

```
<ObProgram name="commonNavBar">
```

The stylesheet name identifies the base stylesheet that is associated with the `program` specified by `ObProgram`.

```
<ObStyleSheet name="stylesheetname.xsl"/>
```

For example:

```
<ObStyleSheet name="gsc_front.xsl"/>
```

The following element identifies the XML schema file that is associated with the program specified by ObProgram:

```
<ObSchema name="XML_schema_name.xsd"/>
```

For example:

```
<ObSchema name="gsc_front.xsd"/>
```

Note: There is only one `ObStyleSheet` element and only one ObSchema element for each ObProgram element.

The element below identifies an ObButton element that is associated with the program specified by ObProgram. There may be anywhere from zero to many ObButton elements for each ObProgram element:

```
<ObButton name="a_button_name"/>
```

For example, in the program name for save, the first button is:

```
<ObButton name="groupSubscribe"/>
```

An ObButton is a COREid-specific construct that packages a graphic, mouseover text, and link as a unit to be built into the page.

You *cannot* change the content of the named button package that is provided in the Output XML. You can, however, remove the entry for the button from the registration file, in which case it does not appear in the finished page. And you can control where and whether the button is displayed using the XSL stylesheet.

Important: Oblix recommends that only experienced developers consider editing the registration file. Use *extreme care* if you decide to edit a registration file.

Excerpt: User Manager Registration File

Following is an excerpt from the User Manager registration file in *COREid_install_dir*\identity\oblix\apps\userservcenter\bin\userservcenterreg.xml with information for the view program highlighted in **bold**.

Note: The XML schema file name that applies to the OutPutXML data for the view program is usc_profile.xsd and the stylesheet file to be used is usc_profile.xsl. In fact, the usc_profile.xsl stylesheet is used by several functions. For details about this stylesheet, see "XSL Stylesheet Content" on page 36.

```
<?xml version="1.0"?>
<ObProgramRegistry>
 <ObApplication name="userservcenter">
     <ObProgram name="front">
       <ObStyleSheet name="usc_profile.xsl" />
       <ObSchema name="usc_front.xsd" />
     </ObProgram>
     <ObProgram name="commonNavbar">
       <ObStyleSheet name="usc_profile.xsl" />
       <ObSchema name="usc_front.xsd" />
     </ObProgram>
   ...

     <ObProgram name="deactivateUserArchive">
     <ObStyleSheetname=
       "Deactuser_purgearchiveconfirm.xsl"/>
     <ObButton name="wfArchivePurgeBack"/>
     <ObSchema name="usc_deactivateUserPurge.xsd"/>
     </ObProgram>
     <ObProgram name="view">
       <ObStyleSheet name="usc_profile.xsl"/>
       <ObButton name="initiateDeactivateUser"/>
       <ObButton name="userreactivate"/>
       <ObButton name="userModify"/>
       <ObSchema name="usc_profile.xsd"/>
     </ObProgram>
     <ObProgram name="modify">
       <ObStyleSheet name="usc_profile.xsl"/>
       <ObButton name="userSaveChange"/>
       <ObButton name="userCancelChange"/>
       <ObSchema name="usc_profile.xsd"/>
     </ObProgram>
```

```
    ...
  </ObApplication>
</ObProgramRegistry>
```

JavaScripts

Because the JavaScripts are supplied by WebPass in direct response to requests from the browser, they are located in directories associated with WebPass, rather than with the COREid Server. For example:

WebPass_install_dir\identity\oblix\lang\shared

For more information, see “Directory Structure” on page 39.

HTML created by PresentationXML has embedded within it references to JavaScript files and functions within the files. A few of these files are associated with specific applications, but most of them are provided under WebPass.

A list of some of the most important of these files, and functions available within them, is provided at “JavaScript Library” on page 69.

Styles

The manner in which COREid presents information, its *style*, is a direct result of the appearance of the graphical images used and the way in which they are combined on the page. Stylesheets control the way in which the images are combined. The images themselves can be changed, or different image names can be used in the stylesheets.

New Stylesheet Structure

Messages in stylesheets depend upon a language. With the NetPoint multiple-language capability, messages have been brought out of the stylesheets and defined separately as variables in msgctlg.xsl (and msgctlg.js for JavaScript files). For details about message catalogs, see “Modifying Catalog Files” on page 141.

In addition, each stylesheet has a corresponding language-specific thin wrapper in:

COREid_install_dir\identity\oblix\lang*langTag*\style0

Each wrapper in \style0 *includes* the main language-neutral stylesheet stored in \shared:

COREid_install_dir\identity\oblix\lang\shared

The purpose of this new thin wrapper is to segregate the main functionality of the stylesheet template, which is language independent, from language-specific messages in the stylesheets.

Example: basic.xsl wrapper stylesheet

For example a typical wrapper stylesheet, basic.xsl, is located in \style0 and may be in your custom style directory:

COREid_install_dir\identity\oblix\lang\en-us\style0\basic.xsl
COREid_install_dir\identity\oblix\lang\en-us*Custom*\basic.xsl

basic.xsl content is shown below:

```
<?xml version="1.0" ?>
- <!--    Copyright (c) 1996-2001, Oblix Inc. All Rights Reserved.
  -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:oblix="http://www.oblix.com/">
  <xsl:include href="./style.xsl" />
  <xsl:include href="../msgctlg.xsl" />
  <xsl:include href="../../shared/basic.xsl" />
  </xsl:stylesheet>
```

The basic.xsl wrapper stylesheet includes a pointer to:

- msgctlg.xsl, one directory up from your custom directory (in identity\oblix\lang\en-us)
- style.xsl file in your custom directory

  Due to the change in location of all image files, a new gifPathName variable is defined in style.xsl. For more information, see “Images” on page 38.
- basic.xsl in identity\oblix\lang\shared.

For more information, see “XSL Stylesheet Content” on page 36.

XSL Stylesheet Content

XSL stylesheets follow a standard developed by the World Wide Web Consortium. See page 171 for information on the standard, and an introduction to XSL syntax.

As discussed earlier, PresentationXML uses the name of the current program as an index to its registration file, to get the name of an associated XSL stylesheet. For example, if the User Manager application is running the view program, then it locates and uses the usc_profile.xsl file as its stylesheet.

The following is partial content of the User Manager usc_profile.xsl stylesheet file. The information is shown in a compressed format in which each occurrence of <xsl:template represents many lines of XSL which for the sake of clarity are *not* shown here:

```
<?xml version="1.0" ?>
<!-- Copyright ... -->
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:oblix=”http://www.oblix.com/”>
<xsl:include href=./basic.xsl" />
<xsl:include href=./selectorinfo.xsl" />
<xsl:include href=./usc_searchform.xsl" />
<xsl:include href=./usc_navbar.xsl" />
<xsl:template match="/">
...
<xsl:template match="oblix:ObProfile"> ...
<xsl:template match="oblix:ObProfile/Oblix/ObForm"> ...
<xsl:template match=”/oblix:Oblix/oblix:WfActorComment”>
<xsl:template match=”oblix:WfActorComment/oblix:ObForm”>
<xsl:template match="oblix:ObHeaderPanel">...
<xsl:template match="oblix:ObPanel">...
<xsl:template match="oblix:ObAttribute"> ...
<xsl:template name =”horizontalButton”> ...
</xsl:stylesheet>
```

The lines beginning with xsl:template match are the top level of a great many lines of XSL instructions, called a *template*, which tell the XSL Transformer how to build the output page.

For example, the line:

```
<xsl:template match="oblix:ObProfile">
```

in effect tells the transformer to look for an occurrence of the oblix:ObProfile element in the OutPutXML and, if it finds it, to begin to apply the instructions following this line in the stylesheet.

In the case of this example stylesheet, building of the HTML begins with the line <xsl:template match="/">. Within that template, the Navigation Bar and the Search Form are built. The remaining templates go on to build the Profile information, including the Header Panel (the one including the user photo) and each of the other panels.

The lines beginning with xsl:include, shown in bold, indicate other XSL files which are to be added in-line to generate the complete XSL stylesheet. These included files can contain references to other included files, and so on. Certain base stylesheet files are used in common by almost all of the applications.

Base stylesheets include:

- **basic.xsl**—Contains templates to define attributes and status and control display information. Contains references to the font.xsl and title.xsl files.
- **searchform.xsl**—Contains templates to create the Searchform line.

- **navbar.xsl**—Contains templates to create the Navigation Bar.
- **font.xsl**—Contains templates to define HTML size, fonts and colors for recurring text such as page headings.
- **title.xsl**—Contains the default titles for the HTML pages.

Each base stylesheet includes other nested stylesheets, and so on. For easy reference, tables giving the names of the most commonly used stylesheets, the names of the templates contained within them, and a brief description of what each template does, are provided in “Stylesheets” on page 47.

“Customizing the NetPoint GUI” on page 85 provides an introduction to the use of XSL stylesheets.

Images

HTML created by PresentationXML has embedded within it references to images. NetPoint supplies its own set of GIF images and supports any type of image that can be read by a browser. Because the images are supplied by WebPass in direct response to requests from the browser, they are located in directories associated with Webpass, as described in “Directory Structure” on page 39.

See “Image Library” on page 67 for the naming convention for these files. For details about incorporating custom images or styles after upgrading to NetPoint 7.0, see the *NetPoint 7.0 Upgrade Guide*.

gifPathName and jsPathName Variables

Due to the change in location of all image files, a new *gifPathName* variable is defined in style.xsl. In addition to style.xsl, the following file also includes the *gifPathName* variable to mention the path for image locations:

install_dir\oblix\lang*langTag*\msgctlg.js

For more information about msgctlg files, see “Modifying Catalog Files” on page 141.

Note: Stylesheets refer to the *gifPathName* variable to locate the image directory. JavaScript files refer to the *jsPathName* variable.

A language independent stylesheet picks up the images from the modified image path mentioned by the *gifPathName* variable, which is important for two reasons:

- It prevents hard-coding of URLs in the stylesheets and makes it easier to re-use the same stylesheet across styles.When customizing stylesheets, you should use this global variable whenever constructing a URL path to a GIF.

- It incorporates the current language and current style tag and generates the correct path.

Example—style.xsl with variables highlighted

The style.xsl wrapper resides in \style0 and can reside in your custom directory:

COREid_install_dir\identity\oblix\lang\en-us\style0\style.xsl
COREid_install_dir\identity\oblix\lang\en-us*Custom*\style.xsl

A sample style.xsl is shown below.

```
<?xml version="1.0" ?>
- <!--   Copyright (c) 1996-2002, Oblix Inc. All Rights Reserved.
  -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:oblix="http://www.oblix.com/">
  <xsl:variable name="styleName">style0</xsl:variable>
  <xsl:variable name="localeName">en-us</xsl:variable>
- <xsl:variable name="gifPathName">
  ../../../lang/
  <xsl:value-of select="$localeName" />
  /
  <xsl:value-of select="$styleName" />
  </xsl:variable>
  <xsl:variable name="jsPathName">../../../lang/shared</xsl:variable>
  </xsl:stylesheet>
```

PresentationXML Libraries

This section treats several of the components described earlier as parts of a library of information used to implement PresentationXML as if it were a programming language. It provides more detail and pinpoints the location of the files within NetPoint directories.

Directory Structure

NetPoint can be installed starting at the root directory or in a specified subdirectory. For example:

Default on Windows—C:\Program Files\NetPoint\identity
Default on Unix—/opt/netpoint/identity
In this manual—*COREid_install_dir*\identity

Prior to NetPoint 6.5, the PresentationXML library was provided under two directories and distributed depending upon how the files were likely to be used.

NetPoint 6.5 and later versions include Language Packs that enable you to display static information to users in their native language. English is the default language for which no Language Pack is required. All NetPoint installations include a directory named with the en-us language tag (*langTag*).

When you install additional Language Packs you will see other *langTag* directories. For example, a French Language Pack results in a directory named *fr-fr*. For details about installing Language Packs, see the *NetPoint 7.0 Installation Guide*.

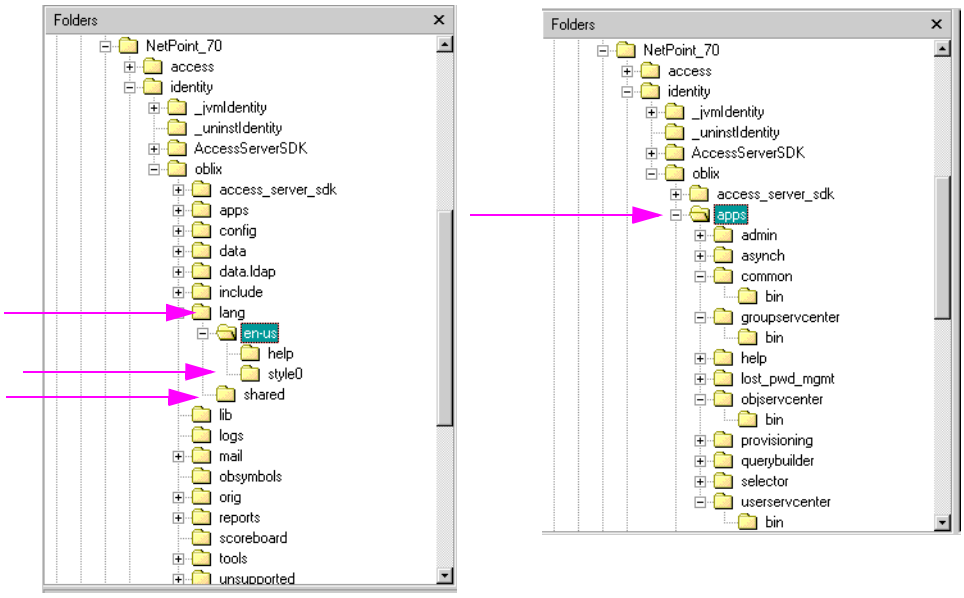
With NetPoint 6.5 and later versions, the directory structure and location of specific files will differ from previous versions of NetPoint. The default directory structure for latest NetPoint PresentationXML Libraries is summarized below:

COREid_install_dir\identity\oblix\apps*AppName*\bin
COREid_install_dir\identity\oblix\lang*langTag*
COREid_install_dir\identity\oblix\lang*langTag*\style0
COREid_install_dir\identity\oblix\lang\shared

WebPass_install_dir\identity\oblix\lang*langTag*
WebPass_install_dir\identity\oblix\lang*langTag*\style0
WebPass_install_dir\identity\oblix\lang\shared
WebPass_install_dir\identity\oblix\WebServices\XMLSchema

Figure 1 shows a default COREid Server Directory structure. Notice the \en-us directory and the \shared directory reside at the same level in the \lang directory. Also notice that the application-specific directories reside under \apps.

Figure 1 Sample Default COREid Server Directory Structure



See “Directory Content” on page 41 for more information.

Directory Content

The contents of default COREid and WebPass directories are introduced here.

The contents of the default COREid directories identified above are outlined in Table 1. See the full list of applications on page 26, in the *AppName* column.

Table 1 Default NetPoint 6.5 COREid PresentationXML Libraries

Default COREid Directories	Contents
COREid_install_dir\identity\oblix\apps*AppName**bin* where *AppName* can be common, groupservcenter, objservcenter, userservcenter, and so on.	• Registration and parameter files specific to the application. **Note**: Dynamically-loadable code for applications is included in COREid Server executables.
COREid_install_dir\identity\oblix\lang*langTag* where *langTag* represents an installed language, such as en-us (English) or fr-fr (French).	Message files for various applications.
COREid_install_dir\identity\oblix\lang*langTag*\style0	• XSL stylesheets for applications point to templates in \shared • Common NetPoint images
COREid_install_dir\identity\oblix\lang\shared	XSL stylesheet templates for various applications

The contents of the default WebPass directories identified earlier is outlined in Table 2.

Table 2 Default NetPoint 6.5 WebPass PresentationXML Libraries

Default WebPass Directories	Contents
WebPass_install_dir\identity\oblix\lang*langTag*	Contains message files for various applications.
WebPass_install_dir\identity\oblix\lang*langTag*\style0	• Image files used in presenting the page. • Copies of style0 stylesheets for client-side processing only.
WebPass_install_dir\identity\oblix\lang\shared	• JavaScripts • Copies of stylesheets for reference only.
WebPass_install_dir\identity\oblix\WebServices\XMLSchema	Contains XML schema files for specific applications.

The differences between content of the COREid and WebPass PresentationXML directories are outlined in Table 3.

Table 3 Differences Between COREid and WebPass Directory Content

Summary of Differences
The XMLschema directory has no effect on PresentationXML operation and exists only on WebPass.
JavaScripts and image files are always provided by WebPass Due to the change in location of all image files, a new gifPathName variable is defined in \identity\oblix\lang\langTag\style0\styles.xsl. A language independent stylesheet picks up images from the modified image path mentioned by the variable. Wrapper stylesheets include the styles.xsl stylesheet file. styles.xsl also contains the path for JavaScripts.
Parameter, message, and registration files are used only by COREid applications, as part of the process that generates the OutPutXML.
Stylesheets are available under both COREid Server and WebPass because they can be used in either: a) Server-side processing (the COREid Server builds the completed XSL stylesheet). b) Client-side processing (WebPass assists the client browser to build the final stylesheet by sending it pieces of other stylesheets to be included in-line).

For example:

Before NetPoint 6.5—*COREid_install_dir*\identity\oblix\apps*AppName*
Before NetPoint 6.5—*WebPass_install_dir*\identity\oblix\apps*AppName*

where *AppName* represents Group Manager (groupservcenter), Org. Manager (objservcenter), User Manager (userservcenter), and the like.

The *AppName* directory included include three significant subdirectories:

Before NetPoint 6.5—*AppName*\bin
Before NetPoint 6.5—*AppName*\ui
Before NetPoint 6.5—*AppName*\xmlschema

Note: NetPoint includes Language Packs that allow you to display static information to users in their native language. English is the default language for which no Language Pack is required. As a result, the directory structure and location of specific files has changed.

Following discussions provide more information on the content of the directories identified above.

COREid_install_dir\identity\oblix\apps*AppName*\bin

The *AppName* directory refers to specific applications, including:

common
groupservcenter
objservcenter
userservcenter
and others

Each *AppName*\bin directory contains a registration file, as discussed in "Registration Files" on page 31. In addition, each *AppName*\bin directory contains a small number of XML files that contain parameters for the specific application. All application-specific subdirectories include the same types of information. Although, the userservcenter\bin directory also includes several certificate files.

For example, in groupservcenter\bin, you will find the following files:

- groupservcenterreg.xml—The registration file for the Group Manager application.
- groupservcenterparams.xml—This file, and other files ending in params.xml, contain parameters that are used to guide execution of the application:
  - gsc_wf_params.xml—Create workflow, remove workflow, change attribute workflow parameters. The files below duplicate this information and are needed for special handling required for the IBM SecureWay directory server:

    gsc_wf_params-base.xml
    gsc_wf_params-sw.xml
- gsc_wfqs_params.xml—Create group basic template parameters.

- gscaclparams.xml—Example that shows a set of parameters and their value ranges for simple access control. Sample data is included that is not meant to be used in any deployment.

Note: *AppName*.dll files, available before NetPoint 6.5, contained logic the COREid Server executed for each application. However, now application module logic and libraries are compiled into the COREid Server executable.

See “NetPoint Parameter Files” on page 193 for a list of all parameter files whose content is designed to be changed by the user.

Important: Do *not* change the content of any of these files *unless* the named file is listed in “NetPoint Parameter Files” on page 193 *and* the parameter is described there *and* the description does not forbid you to change the parameter.

COREid_install_dir\identity\oblix\lang*langTag*

To support multiple languages, NetPoint provides a specific named directory for each installed language. For example, \lang\en-us contains English language files; \lang\fr-fr contains French language files, and so on. Both the default NetPoint style directory and any custom style directories you create are stored within each installed language directory.

COREid_install_dir\identity\oblix\lang\en-us*NewStyle*
COREid_install_dir\identity\oblix\lang\en-us*style0*

COREid_install_dir\identity\oblix\lang\fr-fr*NewStyle*
COREid_install_dir\identity\oblix\lang\fr-fr*style0*

Each *langTag* directory also contains message files for various applications in a specific language, such as English. A \help directory is also stored here.

You may customize messages in each *langTag* directory by editing specific msg.xml files. See “Customizing NetPoint” on page 80 for more information on the method to customize a stylesheet.

COREid_install_dir\identity\oblix\lang*langTag*\style0

The \style0 directory within each *langTag* directory provides default wrapper stylesheets that are specific to each application. For example:

gsc_ is the prefix for Group Manager wrapper stylesheets.
osc_ is the prefix for Org Manager wrapper stylesheets.
usc_ is the prefix for User Manager wrapper stylesheets.
wf_ is the prefix for workflow wrapper stylesheets.
There are others.

Each default wrapper stylesheet points to default global stylesheets in the \shared directory. In addition, several common NetPoint image (GIF) files are stored here. When you add a style using the Configure Styles function in the COREid System Console, and copy from the Classic Style, the contents of \style0 are duplicated in your custom directory.

When customizing a style for NetPoint, you may overwrite a default wrapper stylesheet in your custom directory with a copy of the \shared stylesheet you intend to customize. No wrapper stylesheet in your custom directory should inherit from (or reference) a default global stylesheet in the \shared folder.

Note: Oblix recommends that you retain the files in \style0 as they are in case you need to revert to the default style. See “Customizing NetPoint” on page 80 for more information.

The giflist.xml file included in previous versions of NetPoint, is not available.

COREid_install_dir\identity\oblix\lang\shared

The \shared directory contains default global stylesheets that apply to various applications in all languages.

You may edit stylesheets in \shared to institute a global change for all languages. However default stylesheets in \shared, and in \style0, will be updated periodically by Oblix. A customized stylesheet in \shared or \style0 may be overwritten during product patches or upgrades. Other stylesheets may depend on the updates making it risky to overwrite an updated default stylesheet with the back up copy of a customized style. A better practice is to copy a default stylesheet from \shared into your custom directory, then customize the copy.

Note: Oblix recommends that you retain the files in \shared as they are in case you need to revert to the default style.

WebPass_install_dir\identity\oblix\lang*langTag*

This directory contains message catalog files for various applications but is *not* an exact duplicate of the *langTag* directory on the COREid side:

- Copies of several message files
- Additional message files, such as webpassmsg.xml and others
- HTML files such as ldap_personoc.html

This directory also includes a help directory, and a docs directory for Web server related files. Again, you may edit message files here and propagate changes to other WebPass hosts. See “Customizing NetPoint” on page 80 for more information.

WebPass_install_dir\identity\oblix\lang*langTag*\style0

Contains copies of default XSL wrapper stylesheets for various applications from *COREid_install_dir*\identity\oblix\lang*langTag*\style0.

Also included are the image files used by NetPoint when presenting the page, which are always provided by WebPass as direct responses to browser requests.

Note: NetPoint relies on the images provided in this directory. Oblix recommends that you do not alter default images.

After you add a style to NetPoint and customize stylesheets, you will create the same style-related directory structure on the WebPass. You must copy all images you intend to use into your custom directory structure on WebPass. Even if you have made no changes to images you must still:

Copy Images From—*WebPass_install_dir*\identity\lang\en-us\style0
Copy Images To—*WebPass_install_dir*\identity\lang\en-us*CustomStyle*

As with other default style files, images may be updated by Oblix periodically.

WebPass_install_dir\identity\oblix\lang\shared

This directory contains default global files that WebPass uses in response to requests:

- JavaScripts used by WebPass
- Copies of XSL stylesheets in *COREid_install_dir*\identity\oblix\lang\shared directory for use with client-side processing.

As with other default style files, these may be updated by Oblix periodically. Oblix recommends that you do not alter default files.You may copy and customize JavaScripts using the same methodology as if modifying a stylesheet on the COREid side. See “Customizing NetPoint” on page 80.

WebPass_install_dir\identity\oblix\WebServices\XMLSchema

This directory contains the XML schemas, as xsd files, that define elements specific to various applications. For example

gsc_... identifies Group Manager files.
osc_... identifies Org Manager files.
usc_... identifies User Manager files.
workflow... identifies Workflow files.

The XMLSchema has no affect on PresentationXML and is included only on WebPass.

As with other default style files, these may be updated by Oblix periodically.

The NetPoint system relies on these files. Oblix recommends that you retain the files in this directory as is.

Stylesheets

As discussed above, the default XSL stylesheets and wrapper stylesheets for various NetPoint applications can be found in the following directories:

COREid_install_dir\identity\oblix\lang*langTag*\style0 -- *wrapper stylesheets*
COREid_install_dir\identity\oblix\lang\shared -- *stylesheets*

WebPass_install_dir\identity\oblix\lang*langTag*\style0
WebPass_install_dir\identity\oblix\lang\shared

On WebPass, copies are included for client-side processing if that method is used.

The following base stylesheet files provide a foundation for all other stylesheets and contain relative pointers to other stylesheets:

- **basic.xsl**—Provides templates to define attributes and status and control display information, including references to the font.xsl and title.xsl files. For more information, see “basic.xsl” on page 48.
- **font.xsl**—Contains templates to define HTML size, fonts and colors for recurring text such as page headings. For more information, see “font.xsl” on page 53.
- **title.xsl**—Contains the default titles for the HTML pages. For more information, see “title.xsl” on page 55.
- **navbar.xsl**—Provides the template to define the NetPoint Navigation Bar. For more information, see “navbar.xsl” on page 55.
- **searchform.xsl**—Provides templates to create the Searchform line. The following searchform.xsl files appear as well, and all point to searchform.xsl:

  gsc_searchform.xsl
  osc_searchform.xsl
  usc_searchform.xsl

  For more information, see “searchform.xsl” on page 56.

Important: Oblix periodically updates stylesheets and recommends retaining default stylesheets to serve as a predictable stylesheet library.

---

basic.xsl

The basic.xsl wrapper stylesheet in the \lang\en-us\style0 or your custom directory provides references to style.xsl, msgctlg.xsl, and to the basic.xsl stylesheet located in the \shared subdirectory.

The basic.xsl file in the following directories includes the variables in Table 4 and the templates in Table 5.

COREid_install_dir\identity\oblix\lang\shared
WebPass_install_dir\identity\oblix\lang\shared

Table 4 Variables Defined in basic.xsl

textSLength	The default length for a textbox.
textSMaxLength	The maximum length for a textbox.
namePrefix	Oblix internal, do not change.
singlequote	Defines a constant for the single quote character. This is specified as a character that needs to be handled in a special manner and is required for literal strings in JavaScript code enclosed in single quotes.
charsToEscape	Used in the `PrepForJS XSL` template. This is a list of characters that need to be escaped with &.

Table 5 identifies actual templates available in basic.xsl.

Table 5 basic.xsl Templates

`oblix:`ObDisplay	Each Oblix display type (for example checkbox, textbox, bitstring, and so on) is nested within the oblix:ObDisplay element. This matching template, sometimes called the Dispatcher, calls the corresponding display type template to properly generate the HTML for that display type. Additional HTML logic is added to properly include the + or - button in modify mode.
oblix:ObBitString	Generates the bitstring display type data as HTML text in view mode or as an HTML textbox in modify mode.
oblix:ObBoolean	Generates the Boolean display type data as text strings in view mode or radio buttons in modify mode.
oblix:ObCheckBox	Generates the checkbox display type data as text strings in view mode or an HTML checkbox in modify mode.

Table 5 basic.xsl Templates

oblix:ObDate	In view mode this template generates the date display type as text strings, using a format corresponding to the date type specified (ObMonthDYDate, ObDMYDate, for example). In modify mode this template generates the date display type as an HTML select box (one for year, one for month, and one for day). This template calls the ObDateValue template to generate the actual select boxes.
oblix:ObDateValue	In view mode this template generates one entered value of the date display type data as a text string with a format corresponding to the date type specified (for example ObMonthDYDate, ObDMYDate). In modify mode it generates an HTML select box (one for year, one for month, and one for day).
oblix:ObDn	In both view and modify mode, generates the dn display type as hyperlink text. The dn value will also be prepended with an image, if one is supplied from the XML.
oblix:ObEmail	In view mode generates the email display type as hyperlinked text in the form mailto:<email address>). In modify mode, this is an HTML text box.
oblix:ObFacsimile TelNum	In view mode, this template generates the facsimileTelNum display type data as a text string with the fax value first, followed by optional parameters describing whether it is TwoDimensional, FineResolution, b4Length, a3Width, b4Width, uncompressed, and unlimitedLength. In modify mode, the text string is displayed as an HTML textbox and the parameter properties are displayed as checkboxes.
oblix:ObGeneric Selector	Generates the generic selector display type data as hyperlinked text in both view and modify mode. If the data is for a user, a user image is prefixed. If the data is for a group, a group image is prefixed. If the data is an object, an object image is prefixed. In addition, the modify mode also generates the selector button.
oblix:ObGif	Generates the GIF display type data as an image using the src, width, height, and alt information from the XML, in both view and modify mode. In modify mode, a file upload box is also generated.
oblix:ObGifUrlText	A named template that is called by ObGifUrl. Used only in modify mode to generate the gifurl information as a textbox with the specified length and maxlength.

Table 5 basic.xsl Templates

oblix:ObGifUrl	In both view and modify mode, generates the gifurl display type data as an image using ObImage's XML attributes obhref (the hyperlink), obalt (the alternate text), obwidth (the width of the image), and obheight (the height of the image). In modify mode, an additional text box is generated by calling the template ObGifUrlText.
oblix:ObLocationDn	Generates the location dn display type data as a link by calling the ObLink template.
oblix:ObMedia	Generates the media display type data as a hyperlink with an image, if one is supplied or the specified display name, if one is supplied. For modify mode, an additional file browse button is also generated.
oblix:ObPassword	In modify mode, generates an HTML password input box with the specified length and max length. If oboldpswd is true, then the old password is also prompted for, using a password input box. (This element is not used in view mode).
oblix:ObPostal Address	In view mode, generates the postal address display type value as a text string. In modify mode, the values are presented as modifiable data in text boxes.
oblix:ObQuery Builder	In modify mode, each value is displayed as modifiable data in a textbox. A querybuilder button is also generated. (This element is not used in view mode).
oblix:ObRadio	In view mode, generates the radio display type values as text strings. In modify mode, the values are generated as HTML radio buttons.
oblix:ObSelect	In view mode, generates the select display type values as text strings. In modify mode, the values are generated as HTML select boxes.
oblix:ObSMIME Certificate	Generates the SMIMECertficate display type as text strings.
oblix:ObTextM	In view mode generates the textM display type value as a text string. In modify mode the values are generated as HTML multi-line text boxes with the specified columns and rows.
oblix:ObTextS	In view mode, generates the textS display type value as a text string. In modify mode, the values are generated as HTML single-line text boxes with a specified maxlength and length.

Table 5 basic.xsl Templates

oblix:ObNumericStr	In view mode, generates the numeric string display type value as a text string. In modify mode, the value is generated in a text box with the specified maxlength and length, along with a JavaScript validation to ensure that the text field only accepts numeric values.
oblix:ObValue	Generates the PC Data text in the Oblix:ObValue element.
oblix:ObApplet	Generates an HTML applet. Information includes name, codebase, code, width, height, align, target, and archive. If the applet requires input parameters, they are specified in the oblix:ObParam element.
oblix:ObScripts	Calls the oblix:ObScript template for each ObScript element within the ObScripts element.
oblix:ObScript	Generates the script tag to allow the JavaScript specified in the Oblix:ObScript element to be referenced within the HTML.
oblix:ObButton	Generates a button, either a hyperlink text or image, using the information provided in the XML. Information includes the href, mouse over, and optional image. If an image is not specified the anchor text is displayed.
oblix:ObInput	Generates the HTML input tag using the specified information in the XML element.
oblix:ObLink	In view mode generates a hyperlink with the specified href and mouse over values, along with either an image or a text string. In modify mode a text box is generated.
oblix:ObImage	Generates an image using the specified obwidth, obheight, obalt, and href values.
requestLessValue	A called template that generates the proper HTML tag to enable the "request a ticket to remove an attribute" functionality.
requestMoreValue	A called template that generates the proper HTML tag to enable the "request a ticket to modify an attribute" functionality.
outputDateChoices	A called template that generates the date choices for modify mode.
ObDateMonth	A called template that generates the month select box in modify mode.

Table 5 basic.xsl Templates

ObDateDay	A called template that generates the day select box in modify mode.
ObDateYear	A called template that generates the year select box (by default, 1993 - 2012) in modify mode.
ObDateHourOption	A called template that generates the hour select box (00 - 23) in modify mode.
ObDateTZHourOption	A called template that generates the hour select box (00 - 12) in modify mode.
ObDateMinOrSec Option	A called template that generates the minute or the hour select box (00 - 59) in modify mode.
requiredAttrInput	A called template that generates hidden HTML input tags, intended for Oblix internal purposes. This is called by the Oblix:ObDisplay template.
moreValue	A called template that generates the proper HTML tag to enable the “add more values” functionality (The + button).
lessValue	A called template that generates the proper HTML tag to enable the “remove a value” functionality (The - button).
oblix:ObDisplay Properties	A called template that adds the onX event which can be any of: onChange, onClick, onDblClick, onDragStart, onFilterChange, onHelp, onKeyDown, onKeyPress, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onSelectStart, or disabled.
oblix:ObError	Generates the error message.
oblix:ObText Message	Generates the specified text message.
MakeHidden ObAttribute	A called template that generates hidden HTML input tags, intended for Oblix internal purposes.
ObDisplayPretty Print	A called template that generates values in a table format.
ObLinkPrettyPrint	A called template that generates link values in a table format.
ObLinkModifyPrettyPrint	A called template that generates each link values as a checkbox in a table format.

Table 5 basic.xsl Templates

Makeobvarname	A called template that converts the input parameter from x to Obx, changing hyphens to underscores.
AddLineBreaks	A called template that adds line breaks to a string, in the places where the string contains
oblix:PrepForJS	A called template that escapes characters in preparation for calling a JavaScript.
oblix:AddJava PluginLayer	A called template that generates an error display if the Java Runtime Environment 1.3 or higher is needed but not available.

font.xsl

This stylesheet contains templates to define HTML size, fonts and colors for recurring text such as page headings. The font.xsl file in the following directories includes the variables in Table 6 and templates in Table 7.

COREid_install_dir\identity\oblix\lang\shared
WebPass_install_dir\identity\oblix\lang\shared

Table 6 Variables Defined in font.xsl

pageHeaderSize	Defines the size of the heading for the page.
pageHeaderColor	Defines the color of the heading for the page.
pageHeaderFont	Defines the font of the heading for the page.
subHeadingSize	Defines the size of the subheading for the page.
subHeadingColor	Defines the color of the subheading for the page.
subHeadingFont	Defines the font of the subheading for the page.
contentTitleSize	Defines the size of the content title (for example the attribute display name in a profile page).
contentTitleColor	Defines the color of the content title.
contentTitleFont	Defines the font of the content title.
contentTextSize	Defines the size of the content text (for example the attribute value in a profile page).
contentTextColor	Defines the color of the content text.
contentTextFont	Defines the font to be used for the content text.
pageWarningSize	Defines the size of the warning message.

Table 6 Variables Defined in font.xsl

pageWarningColor	Defines the color of the warning message.
pageWarningFont	Defines the font to be used for the warning message.
anchorTextSize	Defines the size of the anchor text to be used with login.xsl and logout.xsl. In particular this variable applies to the anchor text to be used for linking to www.oblix.com and www.oblix.com/support.
anchorTextColor	Defines the color of the anchor text for login.xsl and logout.xsl.
anchorTextFont	Defines the font of the anchor text for login.xsl and logout.xsl.

Table 7 identifies the templates in font.xsl.

Table 7 Templates in font.xsl

addPageHeaderAttr	A called template that is used to set the proper color, size, and font for the page header. (for example the word Profile, which is the title of a profile page).
addSubHeadingAttr	A called template that is used to set the proper color, size, and font for the sub heading. (for example the Panel name in a profile page).
addContentTitle Attr	A called template that is used to set the proper color, size, and font for the content title. (for example the attribute display name in a profile page).
addContentTextAttr	A called template that is used to set the proper color, size, and font for the content text. (for example the attribute value in a profile page).
addPageWarningAttr	A called template that is used to set the proper color, size, and font for the page warning.
addAnchorTextAttr	A called template that is used to set the proper color, size, and font for the anchor text created by login.xsl and logout.xsl.

title.xsl

This stylesheet the default titles for the HTML pages. The title.xsl file in the following directories include the variables in Table 8. There are no templates in title.xsl.

COREid_install_dir\identity\oblix\lang\shared
WebPass_install_dir\identity\oblix\lang\shared

Table 8 Variables Defined in title.xsl

corpdirTitle	HTML title for all Publisher pages.
userservcenter Title	HTML title for all User Manager pages.
groupservcenter Title	HTML title for all Group Manager pages.
objservcenterTitle	HTML title for all Organization Manager pages.
querybuilderTitle	HTML title for all Query Builder pages.
selectorTitle	HTML title for all Selector pages.
lpmTitle	HTML title for all Lost Password Management pages.
defaultTitle	HTML title for all common pages shared across all applications.
corpdir	For Oblix internal use only; used to determine the context of the current page.
userservcenter	Oblix internal use only; used to determine the context of the current page.
groupservcenter	Oblix internal use only; used to determine the context of the current page.
objservcenter	Oblix internal use only; used to determine the context of the current page.

navbar.xsl

In the \shared subdirectory, navbar.xsl provides the template to define the NetPoint Navigation Bar. The following files in the \shared subdirectory also point to navbar.xsl: gsc_navbar.xsl, osc_navbarm.xsl, usc_navbar.xsl.

The navbar.xsl file in the following directories include the templates in Table 9. There are no variables in navbar.xsl.

COREid_install_dir\identity\oblix\lang\shared
WebPass_install_dir\identity\oblix\lang\shared

Table 9 Templates Defined in Navbar.xsl

oblix:ObNavbar	Generates the Navigation Bar (Navbar) that appears at the top of every COREid page. The navbar is made up of several significant parts: Logged in user identification, a list of selectable applications (User Manager, Organization Manager, and so on), Help/About/Logout buttons, application functionality (for example My Identity, Reports, and so on), and Tabs (if there are multiple tabs).
oblix:ObApps	Generates the list of selectable applications.
localButton	A called template that generates a button by filling in the HTML information about the href, mouseover, mouseout, and image or anchor text.
oblix:ObApplication/oblix:ObTabs/ oblix:ObButtons	Generates the buttons used for selecting different tabs.
oblix:ObApplication/ oblix:ObFunctions/oblix:ObButton	Generates the buttons used for selecting the different functionalities within the application (for example My Identity).
oblix:ObApplication/oblix:ObTitle/ oblix:ObButton	For Oblix internal use only, used to set the current context.

searchform.xsl

This template was previously distributed but now templates to create the Searchform line are located in searchform.xsl. The following searchform.xsl files appear as well, and all point to searchform.xsl:

gsc_searchform.xsl
osc_searchform.xsl
usc_searchform.xsl

The searchform.xsl file in the following directories include the templates in Table 10. There are no variables in searchform.xsl.

COREid_install_dir\identity\oblix\lang\shared
WebPass_install_dir\identity\oblix\lang\shared

Table 10 Templates Defined in searchform.xsl

oblix:ObSearchForm	Generates the Searchform, which is located directly beneath the Navigation Bar on most pages. The Searchform may contain many SearchRows.

Table 10 Templates Defined in searchform.xsl

oblix:ObSearchForm/ oblix:ObSearchRow	Generates one row of the search functionality. A row consists of 2 select boxes and one text box, usually one for each search request, such as a search for "full name" "that contains" "John".
oblix:ObSearchForm/ oblix:ObAdvanced Search	Generates the search more, search less, and search all buttons if the advanced search capability is enabled.

XML Schema Elements Library

As mentioned earlier, the OutPutXML information generated by each program within each NetPoint application is hard-coded and is not directly changeable by users. The content of the OutPutXML is *not* controlled by the content of the XML schema file. The file is provided only as a developer's aid, to help you design XSL stylesheets to work with the OutPutXML. Oblix recommends that you *not* change any of the XML schema files.

The following is presented for information only. The XMLSchema directory in *WebPass_install_dir*\identity\oblix\WebServices contains several schema files specific to each application. For example:

gsc_administrationMain.xsd ...
osc_administrationMain.xsd ...
usc_administrationMain.xsd ...
and others ...

If you look at these files, you will see that some contain element definitions that exist only within that file, and many contain references to other schema files.

If you trace nested references deeply enough, you will come to the following six files, which are provided under the XMLSchema directory for the common application:

- **navbar.xsd**—Defines the content for the Navigation Bar, the top two lines of each page, including the application name, help and logout buttons, and the various tabs to select other modules within the application.
- **searchform.xsd**—Defines the SearchForm line, the line available on some pages, that starts with the graphic labeled search. It may include additional lines for more complex search combinations.
- **component_panel.xsd**—Defines the content for profiles, the sets of descriptive information for users, groups or organizations.
- **component_basic.xsd**—Defines many of the lowest level elements, and includes displaytype.xsd and error.xsd.
- **displaytype.xsd**—Defines formatting for each of the NetPoint display types.

- **error.xsd**—Defines the ObError element.

The rest of this section describes the elements that are defined within each of these files.

Important: It is strongly recommended that you leave the content of these files untouched, to serve as a predictable template library.

displaytype.xsd

The displaytype.xsd file in *WebPass_install_dir*\identity\oblix\WebServices\XMLSchema contains the elements described in Table 11. All elements in this file are in the Oblix namespace.

Table 11 displaytype.xsd Schemas

ObDisplay	An element that contains information about a value (or set of values) being generated. This element usually maps to one of the display types such as email or date. This element may contain two children: the element ObDisplayProperties, which is optional, and a required element whose name maps to the display type (for example, ObCheckBox or ObDate). This element contains the following required attributes: **obname** —The name, usually the attribute name, for the value being generated. **obdisplayName**—The display name/user friendly name/printed name. **obdisplayType**—The display type for this value. **obsemanticType**—The semantic type for this value). **obmode**—One of the list: view, modify, or plain. **obshowLabel**—True or false, indicating whether to show the display name when outputting this value. **obrequired**—True or false, indicating whether this is a required value, meaning there must be at least one value entered by the user in modify mode). **obcardinality**—SingleValued or multiValued, indicating whether this is a single-valued or multi-valued attribute. **obcanRequest**—True or false, indicating whether the user can change this value in modify mode.
ObDisplay Properties	Contains zero or more ObDisplayProperty elements.

Table 11 displaytype.xsd Schemas

ObDisplayProperty	Contains the necessary display property for an onX event, which can be onChange, onClick, onDblClick, onDragStart, onFilterChange, onHelp, onKeyDown, onKeyPress, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onSelectStart, or disabled. The XML attribute obname contains the onX event name and the XML attribute obvalue contains the value for the onX event.
ObBitString	Describes the BitString display type. Contains zero or more ObValue elements, each corresponding to a value. In addition, there are two optional XML attributes: oblength, for the length of the textbox and obmaxLength for the maximum length of the textbox.
ObBoolean	Describes the Boolean display type. Contains a required XML attribute obvalue which contains the value true or false.
ObButton	Describes a button. A button contains six optional XML attributes: **obaction**—The action. **obmouseover**—The mouse over message. **obhref**—The href. **obimageUrl**—The location of the image for the button. **obanchorText**—The alternate text if there is no image. **target**—Provided for future use and currently not used.
ObCheckBox	Describes the checkbox display type. Contains zero or more ObChoice elements.
ObChoice	Describes a choice. Usually nested within a display type that allows a choice, such as ObCheckBox. Contains an optional XML attribute obselected. Possible values for obselected are true or false. The default value when this attribute is not specified is false. If a value is selected, it means that the value is one that has been chosen by the user. If a value is not selected, that means it is a possible choice but not one actually chosen by the user. In addition the ObChoice element contains a required XML attribute obdisplayName corresponding to the display name (the user friendly name) for this choice.

Table 11 displaytype.xsd Schemas

ObDate	Describes the date display type. Contains zero or more ObDateValue elements. In addition, it contains the following optional attributes: obseparator—The string separating the parts of the date. If this is not specified, the default value is /. **obformat**—The format in which the date is to be presented. If not specified, the default value is USStandard. Possible values include USStandardFull, EUSStandard, EUSStandardFull, and USStandard. **obdateType**—The manner in which the date is carried as data. Possible values include ObIntegerDate, ObMDYDate, ObDMYDate, ObDMonthYDate, ObISO8601Date, and ObUnknownDate).
ObDateValue	Describes a date value. Contains the following required XML attributes: **obmonth**—A text string that represents the month. **obday**—A text string that represents the day. **obyear**—A text string that represents the year. In addition an optional XML attribute obbadDate describes whether the given date is malformed. The default value for the obbadDate attribute is false meaning the date is not malformed. Possible values are true and false.
ObDn	Describes the dn display type. Contains zero or more obLink elements.
ObEmail	Describes the email display type. Contains zero or more ObValue elements. In addition, the ObEmail element may contain two optional XML attributes: **oblength**—The actual length of the textbox. **obmaxLength**—The maximum allowed length of the textbox.
ObFacsimileTelNum	Describes the facsimile telephone number display type. Contains zero or more ObValue elements and zero or more ObFaxParam elements.
ObFaxParam	Contains an XML attribute obdisplayName.

Table 11 displaytype.xsd Schemas

ObForm	Contains information for generating a form. Contains zero or more ObInput elements. In addition, it contains: **obname**—The required XML attribute providing the name of the form. **obaction**—An optional XML attribute providing the action to take when the form is submitted. **obenctype**—An optional XML attribute providing the encryption type for the form. **obmethod**—An optional XML attribute specifying the method, either post or get, for this form. The default method is post.
ObGenericSelector	Describes the generic selector display type. Contains zero or more ObLink elements and at most one optional ObButton element.
ObGif	Describes the GIF display type. Contains zero or more ObImage elements.
ObGifUrl	Describes the gifurl display type. Contains zero or more ObImage elements. In addition it contains two optional XML attributes: **oblength**—The length of the textbox **obmaxLength**—The maximum length of the textbox.
ObLocationDn	Describes the location dn display type. Contains zero or more ObLink elements.
ObMedia	Describes the media display type. Contains zero or more ObLink elements.
ObMime	Describes the mime display type. Contains a required XML attribute obtype, and an optional XML attribute obfileExt (used to define file extensions).
ObPassword	Contains zero or more ObValue elements. Additionally, it may contain three optional XML attributes: **oblength**—The length of the textbox. **obmaxLength**—The maximum length of the textbox. **oboldpsw**—Which indicates whether to display information about the old password. Possible values for the oboldpsw attribute are true and false.
ObPostalAddress	Contains zero or more ObPostalValue elements. Contains two optional XML attributes: **oblength**—The length of the textbox. **obmaxLength**—The maximum length of the textbox.
ObPostalSubString	PC Data containing the postal sub string.

Table 11 displaytype.xsd Schemas

ObPostalValue	Contains zero or more ObPostalSubString elements.
ObSelect	Describes the select display type. Contains zero or more ObChoice elements. In addition, it contains two optional XML attributes: **obmultiple**—True or false; the default value is false. **obsize**—The size of the select box.
ObTextM	Describes the textM display type. Contains zero or more ObValue elements. In addition, it may contain three optional attributes: **obrows**—The number of rows for the multi-line textbox. **obwrap**—Indicates whether the text should wrap, and takes the values true and false. **obcols**—The number of columns for the multi-line textbox).
ObTextMessage	PC Data describing a text message. Usage for this element is context dependent. One frequent use is to output an error message.
ObTextS	Describes the textS display type. Contains zero or more ObValue elements and zero or more ObTextMessage elements. In addition, it may contain two optional attributes: **oblength**—The length of the textbox. **obmaxLength**—The maximum length of the textbox.
ObQueryBuilder	Describes the query builder display type. Contains zero or more ObValue elements and an optional ObButton element. In addition, it may contain two optional attributes: **oblength**—The length of the textbox. **obmaxLength**—The maximum length of the textbox.
ObRadio	Describes the radio display type. Contains zero or more ObChoice elements.
ObSMIMECertificate	Contains zero or more ObSMIMEValue elements. In addition, it contains three optional ObButton elements.
ObSMIMEValue	Contains zero or more ObNameValuePair elements and two optional ObButton elements.
ObNameValuePair	Contains two required XML attributes: **obcertinfo**—Which has three possible values: issuer, validfrom, and validto. **obcertvalue**—Which is the value that is described by this ObNameValuePair.

Table 11 displaytype.xsd Schemas

ObScript	Contains information for including a script. The name of the script is described by the required XML attribute obname.
ObScripts	Contains one or more ObScript elements.
ObValue	PC Data containing a value.
ObImage	Contains one required XML attribute: **obhref**—The href. and three optional XML attributes: **obalt**—The alt text. **obwidth**—The width of the image. **obheight**—The height of the image.
ObInput	Contains the information required to generate an HTML input tag. Contains up to three XML attributes: **obtype**—The type of input (required). **obname**—The name of the input (required). **obvalue**—The value of the input (optional).
ObLink	Contains zero or more ObImage elements and zero or more ObMime elements. In addition, it contains up to three XML attributes: **obhref**—The href (required). **obmouseOver**—The mouse over message for the link (optional). **obdisplayName**—The display name for the hyperlink (optional).
ObNumericStr	Contains zero or more ObValue elements. In addition, it may contain two optional XML attributes: **oblength**—The length of the textbox. obmaxLength—The maximum length of the textbox.
ObApplet	Contains information describing an applet. Contains zero or more ObParam elements, each describing a parameter for the applet. In addition, it contains the one optional XML attribute obarchive and the required XML attributes obname, obtarget, obcodeBase, obcode, obwidth, obheight, and obalign, all of which map to information required in the HTML applet tag.
ObParam	Contains the name and value for a parameter, in the form of two required attributes: obname and obvalue.

component_basic.xsd

The component_basic.xsd file in *WebPass_install_dir*\identity\oblix\WebServices\XMLSchema contains the elements described in Table 12. All elements in this file are in the oblix namespace.

Table 12 component_basic.xsd Schemas

ObRequestInfo	Oblix internal use only.
ObStatus	Contains the returned status value, either 0 or 1, for the request.
ObVariableText	Contains one ObDisplay element.
ObAttribute	Contains zero or more ObDisplay elements and an optional ObTextMessage element. In addition, an optional XML attribute obattrName may be used to specify the attribute name.

navbar.xsd

The navbar.xsd file in *WebPass_install_dir*\identity\oblix\WebServices\XMLSchema contains the elements described in Table 13. All elements in this file are in the oblix namespace.

Table 13 navbar.xsd Schemas

ObTabs	Consists of zero or more buttons used for describing the different tabs configured for the current application.
ObNavbar	Describes the navigation bar. It may contain up to 8 optional elements: ObRequestInfo, ObScripts, ObMisc, ObApps, ObForm, ObApplication, ObFunctionsButton, and ObStatus. In addition, a required XML attribute obbgcolor specifies the color of the background for this page.
ObMisc	Describes the buttons that will be used for help, about, and logout.
ObApps	Contains zero or more ObApplication elements. Each ObApps element describes an application, such as User Manager or Organization Manager.

Table 13 navbar.xsd Schemas

ObApplication	Describes the details of a specific application, usually nested within ObApps. Contains any of the following four elements: **ObButtons**—Specifies the URL/button information to get to that application. This information is only filled in if the user is not in that application. **ObFunctions**—Specifies the functions available for for this page. **ObTabs**—Specifies the tabs to be used on this page, if any. **ObTitle**—Specifies the title image for this application.
ObFunctions	Describes the functions available on the current **page. Consists of:** **ObButtons**—For example, non-workflow and administrative functionality such as configure proxy, deactivate user or create user. There are zero or more of these. **ObWorkflowFunctions**—For example, workflow incoming request, workflow outgoing request or workflow ticket search form. **ObAdminFunctions**—Administrative functions, such as delegated administration or workflow definition.
ObAdminFunctions	Describes the administrative functionality such as workflow definition and delegated administration.
ObWorkflow Functions	Describe the workflow functionality such as workflow incoming request, workflow outgoing request or workflow ticket search form.
ObTitle	Consists of zero or more buttons that describe the image that applies to the current application.
ObApplicationFunc	Provides further context defining where the user is in screen navigation. This element is not used unless the user is past first level navigation. An example is the navigation information that will be contained here after the user clicks on "Configuration"

searchform.xsd

The searchform.xsd file in *WebPass_install_dir*\identity\oblix\WebServices\XMLSchema contains the elements described in Table 14. All elements in this file are in the oblix namespace.

Table 14 searchform.xsd schemas

ObSearchForm	Describes a search form which consists of zero or more of the following elements: ObHelpContext, ObRequestInfo, ObScripts, ObForm, ObDisplay, ObButton, ObAdvancedSearch, ObSearchRow, and ObStatus.
ObSearchRow	Describes each row of the search form, consisting of zero or more ObDisplay elements. The SearchRow contains the information required to output a table row which contains 3 fields: a text box, and two select boxes.
ObAdvancedSearch	Indicates if the advanced search capability is enabled through inclusion of and the value set for the optional XML attribute obadvancedSearchOn.
ObHelpContext	PC Data containing information about the context of the page, used for online help.

component_panel.xsd

The searchform.xsd file in *WebPass_install_dir*\identity\oblix\WebServices\XMLSchema contains the elements described in Table 15. All elements in this file are in the oblix namespace.

Table 15 component_panel.xsd schemas

ObHeaderPanel	Describes the Header Panel, which contains a number of LDAP attributes described by zero or more ObAttribute elements. In addition, a panel is described with a number of optional XML attributes, those specified by the Administrator during panel configuration in the System Console. The difference between a Panel and a Header Panel is that a profile page can have a number of panels but at most only one header panel.

Table 15 component_panel.xsd schemas

ObPanel	Describes a Panel, which contains a number of LDAP attributes described by zero or more ObAttribute elements. In addition, a panel is described with a number of optional XML attributes, those specified by the administrator during panel configuration in the System Console. The difference between a Panel and a Header Panel is that a profile page can have a number of panels but at most only one header panel.

error.xsd

The searchform.xsd file in *WebPass_install_dir*\identity\oblix\WebServices\XMLSchema contains the elements described in Table 16. All elements in this file are in the oblix namespace.

Table 16 error.xsd schemas

ObError	Describes an error, including an error message contained in the ObTextMessage element.

Image Library

NetPoint provides default GIF files used in presenting the page and supports any type of image the browser supports. These are always provided by WebPass as direct responses to browser requests and are located in *WebPass_install_dir*\identity\oblix\lang*langTag*\style0, as described in "Directory Structure" on page 39.

Important: NetPoint relies on these images files and Oblix recommends that you copy the files to your custom directory, make changes, then include the updated copies in stylesheets. See "Customizing NetPoint" on page 80. On rare occasions, if your custom image does not appear but the default does, you may need to change the default image.

Most of the filenames conform to a standard naming convention. Understanding this convention will help you avoid referencing the wrong image in your stylesheets. An image name includes three elements: <*ImagePrefix*><*DescriptiveName*><*ImageState*>. For example:

2FTABgeneratereport2.gif

ImagePrefix—In the example above, *2FTAB* is the ImagePrefix. This part of the GIF image name describes the way in which the image is used. See the following table for a list and description of the various values for the ImagePrefix.

DescriptiveName—In the example, *generatereport* is the DescriptiveName. This part of the GIF image name is an attempt to provide a user friendly description of the function, tab, content, or navigation feature with which the image is associated.

ImageState—This part of the GIF image name indicates whether the image shows as active or inactive. Active means that the function, tab, content or navigation feature matches what is currently displayed on the page. Inactive means that it does not; the image represents an alternative that could be selected. The images are usually identical, except that the active image is in a darker color. The difference in naming is:

- **Active Images**—ImageState is set to 2 to represent active images.
- **Inactive Images**—ImageState is omitted for inactive images.

The image for 2FTABgeneratereport2.gif is shown below. This is the active image.

The image for 2FTABgeneratereport.gif, the inactive image, is shown next for comparison.

Table 17 lists the possible ImagePrefix values for image files:

Table 17 ImagePrefix Types

Image Type	ImagePrefix Value	Description
Content Buttons	CBUTTON	Associated with activities that select certain subsets of the available data, such as selecting a certain user or adding data.
Content Buttons (Small)	2CBUTTON	Smaller versions of the same.
Content Image	CIMAGE	Descriptive images, not associated with a choice.
Content Tabs	CTAB	Used to select major categories of data, such as a group type.
Content Tabs (Small)	2CTAB	Smaller versions of the same.
Function Tab	FTAB	Associated with activities that the user wants COREid to perform, such as creating a group, or generating a report.

Table 17 ImagePrefix Types

Image Type	ImagePrefix Value	Description
Function Tabs (Small)	2FTAB	Smaller versions of the same.
Login Screen Images	LOGIN	These are not for customer use.
Page Navigation	NAV	Used to indicate a change to a different screen, usually with no modification to data.
Page Navigation (Small)	2NAV	Smaller versions of the same.

JavaScript Library

Most JavaScript files are located as described in “Directory Structure” on page 39, in *WebPass_install_dir*\identity\oblix\lang\shared.

JavaScript files refer to the jsPathName variable to locate the image directory. For details, see “gifPathName and jsPathName Variables” on page 38.

Language-specific messages are referred to through variables in message catalog files. For additional information, see “Changing Message Catalogs and MouseOver Text” on page 148.

There are too many JavaScript files to document the content of all of them here. However, this section provides a list of functions and what they do, for some of the most frequently used Javascript files. You may find a function in this Library that you want to use at a non-standard place in the stylesheet, or the function might serve as a starting point for a new one of your own.

Important: Again, Oblix recommends that you retain files in \shared as a reliable default. You may, however, copy the files in \shared to a custom directory, change the content in the copy, then write your stylesheets to include the new JavaScript files and functions. See “Customizing NetPoint” on page 80.

Confirm.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, confirm.js includes the functions described in Table 18.

Table 18 Confirm.js

Function Name	Description
myConfirm	Takes a message and a UR as arguments. A pop-up confirmation window will appear with the message. If the user presses okay the browser takes the user to the URL.
confirmDelete	Confirmation of a delete action in the form of a pop-up confirmation window appears. If the user presses okay the browser takes the user to another Web page whose href uses most of the href of the original URL, but replaces the program information with the argument URL.
confirmClear	Outputs a Windows confirmation pop-up box asking if the user wants to clear x where x is the argument name. If so, the browser takes the user to a new URL based on the information provided in the argument URL.

Customizeresults.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, Customizeresults.js includes the functions described in Table 19.

Table 19 Customizeresults.js

Function Name	Description
configureCustomizeresultsFormAnd Submit	Takes no arguments; validates the browse results columns selected before submitting the form. Validation restrictions include requiring at least one attribute to be selected, no attribute shall be selected twice, and selected attributes must not be separated by blank selections.
cancelCustomize resultsFormAnd Submit	Resets the customizeResultsForm such that no validation is performed.

Deactivateuser.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, Deactivateuser.js

includes the functions described in Table 20.

Table 20 Deactivateuser.js

Function Name	Description
startSearchAnd Submit	Submit the monitor search form.
DeactContinue Search	Provides the logic for pressing previous/next in the deactivate search results page. The argument `subprogram` signals whether this is for previous or next. For next, if this is already the last set of search results, an error message pops up saying so. For previous, if this is the first page, an error message indicatse this.
submitresults	Submit the deactivate search form with validation. If validation succeeds the function submits the argument URL to the COREid server.
toggleselect	Toggle the search results form checkboxes from false to true.
sortDeactivated Users(sortBy, sortOrder)	Set the sort information in the deactivate search results page using the argument information.

Groupsubscription.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, Groupsubscription.js includes the functions described in Table 21.

Table 21 Groupsubscription.js

Function Name	Description
resetsubscription	Reset the ObGroupsToSubscribeForm form.
submitresults(URL)	Submit the ObGroupsToSubscribeForm form after formulating the required information for the COREid server.

Helpcommon.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, Helpcommon.js includes the functions described in Table 22.

Table 22 Helpcommon.js

Function Name	Description
ObHelp	Build the help URL by appending to the argument `helpUrl` using the current page's help context.

Table 22 Helpcommon.js

Function Name	Description
SetHelpContext	Set the help context of the current page using the information provided from the arguments.
BuildParameter	Helper function used by `ObHelp` to build the name value pair parameters into the form &name=value.

Horizontalprofile.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, Horizontalprofile.js includes the functions described in Table 23.

Table 23 Horizontalprofile.js

Function Name	Description
toggleImage	Change the image with the argument `imageName` to use the image from the argument srcName. This is a helper function for the `change` function.
change	Change the image specified by the arguments oldName, oldIndex to a new image specified by the argument newName, newIndex.
showPanel	Show the panel with the argument panel id name newPanel and argument panel id index newPanelIdx. In addition, the function hides the current panel.
initDynamicAux Classes	Initializes the values of valuesOldName. Iterates through the elements in profileForm and stores the values of the named element oldName in the array valuesOldName. The argument oldName is the name of the element that stores the initial state of the set of auxiliary classes prior to the request. (The name used is ObOldAuxClasses).
diffAuxClassSet	Determine whether there is a difference in content between firstSet (the first argument) and secondSet (the second argument). Returns true if there is a difference, false otherwise.

Table 23 Horizontalprofile.js

Function Name	Description
isAuxClassChange	Determines whether or not a group type has been newly selected or removed. The current state of group types is stored in ObAuxClasses. The initial state is in ObOldAuxClasses. The method iterates over the elements of the profileForm and stores the values of any named newName elements if selected. It stores these values in valuesNewName. The method then compares valuesNewName to valuesOldName. If there is an element in valuesNewName not in valuesOldName or vice versa, then the state of auxiliary classes has changed. This method is used to determine the toggle state of the save image (either 'save' or 'update'). The argument newName is the name of the element that stores the current state of the set of auxiliary classes (the name used is ObAuxClasses). Returns true if there is a change in the state of the selected group types, false otherwise.
doSaveRequest	Does the save request action. This action can result in either the data being saved, or a request for the same page, if the user selected auxiliary classes whose attributes are not visible in the display. The argument oldName is the name of the element that stores the original state of the auxiliary classes. The argument newName is the name of the element that stores the current state of the auxiliary classes. The argument saveAction is the name of the action that will result in the entry being saved. The argument moreAction is the name of the action that will result in a request for the same page with new attributes.
doToggle	This method toggles the image of the save button if the state of auxiliary classes has changed. It dynamically determines the source of the save button by looking up imageSave in the set of images in the document. It then re-sources the image by assigning a new image source to the save button element. The method looks up the image by comparing the value of its source. The argument oldName is the name of the element that stores the original state of the auxiliary classes. The argument newName is the name of the element that stores the current state of the auxiliary classes. The argument imageSave is the name of the save button image. The argument im`ageMore` is the name of the more button image.

Misc.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, Misc.js includes the functions described in Table 24.

Important: Oblix strongly recommends that you do *not* change mics.js. This is a system level file. It is contained in almost every style sheet as an include file. Errors in this file will affect most every style sheet that the system uses. This becomes an insidious problem to solve. Custcare will not know if this file has been modified, and indeed, the customer may not know either. Changing this file can cause hidden problems, or problems down the road.

Table 24 Misc.js

Function Name	Description
obDetectBrowser	Detects the browser version of the client and set the proper variable.
submitFormAfter Confirm	Submits the form after the argument confirmation message is displayed.
submitLoginForm	Used for submitting the login form. Checks first to ensure the password is entered and then forms the proper URL for the next page - one that will bring the user to the application selected.
checkPasswordEnterKey	Enable feature for allowing users to submit the login form when they press enter (or any other event specified in the argument). (For example, use this if you want to submit the login form when the user presses enter).
onUserType	For login form, reload the page when the user selects a different user type.
onApplication	For login form, if the user selected Publisher and optional authenticated view is disabled, then take the user to Publisher after the application selection.
setFocusToFirst TextElement	Set the focus of the argument form to the first element.
submitForm	Submit the first form in the page.
submitSpecified Form	Submit the form whose name is specified in the argument formName.
submitFormAfter UserAction	Submit the form with the action equal to the argument form. If the form action is activateForm, then additional confirmation messages will be displayed.

Table 24 Misc.js

Function Name	Description
submitFormCheck ProfileAttributes	Function used for checking whether the same profile attribute has been selected more than once. The application's logging policy modification screen uses this.
submitFormCheck CommonLogParams	Function used for checking whether numeric value is entered for log file maximum size.
isInteger	General purpose function to determine if the value entered is an integer.
IsNumeric	General purpose function to determine if the value entered is composed of digits.
isFloat	General purpose function to determine if the value entered is float.
isNonNullInteger	General purpose function to determine if the value entered is composed of integers. The difference between this function and the isInteger function is that this function does NOT return false if the input value is empty.
sendtotop	Set the current page to top.
IsStringOblix Compliant	Check to make sure the argument element is a non-empty string that does not contain , or ; or \. If it does, then an error message using the argument message is displayed.
isEmpty	Check if the argument string is empty or contains whitespace.
denyWithAlert	Pop-up an alert message using the argument message.
checkAndSubmitForm	Submit the form if the argument does not equal "login". If the argument equals "passwd", then time out.
validateInput	Validate the following inputs: **rootDN**—ensure it is not empty. **ldapRootPasswd**—ensure it is not empty. **machineNo**—ensure it is not empty. **portNo**—ensure it is an integer. **searchBase**—ensure it is not empty. If all validation succeeds, pop up a final confirmation box for the change and submit the form.
validateSearch	Ensure that the display field record is a valid integer and that it is not empty.

Table 24 Misc.js

Function Name	Description
checkSearchKey	Submit the form and perform proper validation for the forms deactUserSearchForm, searchForm, ObSearchGroupMembersForm, viewGroupMembers, and monitorsearchform.
validateSearchAnd Submit	Used by start search, more fields and less fields in the search functionality to do submit the form. For start search, validation (of the minimum number of characters for the search criteria values) is performed. For more field and less field, calculations of the number of rows required is calculated and then the form is submitted.
continueSearch	Used by the next and back buttons in the search functionality to submit the form. For next, it ensures that this is not already the last set of records. For back, it ensures that this is not already the beginning of the records.
continueSelector Search	Used by selector search next and back buttons. Submit the form after calculating the startFrom value.
doSearch	Submit the search form after assigning the proper values for sortBy and sortOrder, which are both provided as input arguments.
appendElementsTo BackUrl	Append a set of element_name=value pairs to the backUrl. The argument backUrl is the back URL string. The argument elements is the set of elements to append.
startTrim	Trim the beginning whitespace in the argument string.
endTrim	Trim the trailing whitespace in the argument string.
createBackUrl	Create and return the portion of the back URL that contains the forms elements. Append element/value pairs from all forms in the HTML document including those within layers.
PersonSelector	Used to launch the selector. The argument gotoUrl is the (selector) URL we're about to go to. The argument returnUrl is the URL we need to use to get back to the page that invoked the selector.
QueryBuilder	Used to launch query builder. The argument gotoUrl is the (selector) URL we're about to go to. The argument returnUrl is the URL we need to use to get back to the page that invoked the selector.
sendBookmark	Used for sending the search results as a bookmark.

Table 24 Misc.js

Function Name	Description
validateLicense Keys	Ensures that the form values are not empty. If the form values are empty, pops up an error message saying you must enter the license key.
validateLicense KeysAndSubmit	Ensures that the form values (expected to be the license key values) are not empty and then submits the form.
lostPassword	Invokes the lost password functionality after ensuring that the login field is not empty.
certLicenseMessage	Pops up a message to warn the user to install the certificate management license file before any function is enabled.
installIECert	Calls a Virtual Basic Script function to install an x509 certificate. If we are using Netscape, it will contact the common server and get a stream of binary certificate.
extractCert	Calls a Virtual Basic Script function to install an x509 certificate.
performOCSP	Calls back to the modify profile page to use OCSP (Online Certificate Status Protocol) for certificate online checking. Before the URL is directed, it will pop up an alert to warn the users that the OCSP checking will take a while.
validateDeactivatedUserSearch	Ensures that the display record for the deactUserSearchForm is a non-empty integer.
validateGroup MemberSearchAnd Submit	Validates group member search by calling validateGroupMemberSearch. If this is an update member functionality, a pop up message shows to confirm the changes before the form is submitted.
validateGroup MemberSearch	Do validation for group member search, such as ensuring that the minimum number of characters for each search value is satisfied and that the display record is a non-empty integer.
cancelWorkflow	Terminate the current workflow. Upon confirmation, go to the argument URL.
GetCookie	Retrieve the value of the cookie with the argument name.
setFocusToOKButton	Set the focus to the form element that is equal to javascript:top.close() for Internet Explorer 4 and Netscape 6.

Table 24 Misc.js

Function Name	Description
submitFeedBack	Submit the feedback form.
checkJavaPlugin	Check to make sure that the proper java plug-in is installed.
EnableDetectJava PluginLayer	Enable the layer called "DetectJavaPlugin".
EnableJavaPlugin VersionLayer	Enable the layer called "DetectJavaPluginVersion".
DetectPluginFor Applets	Verify that Java plug-ins are installed.
getParameterValue	Retrieve the value of the parameter in the URL with the argument name.

Miscsc.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, Miscsc.js includes the functions described in Table 25.

Important: Oblix strongly recommends that you do *not* change Miscsc.js. This is a system level file. It is contained in almost every style sheet as an include file. Errors in this file will affect most every style sheet that the system uses. This becomes an insidious problem to solve. Custcare will not know if this file has been modified, and indeed, the customer may not know either. Changing this file can cause hidden problems, or problems down the road.

Table 25 Miscsc.js

SCConfirm	Submit the profile form. The argument value program will be assigned to the profileForm program field by default or, if an argument message is not empty, then upon confirmation from the user through the pop up confirmation message.
SCRequestChange AndSubmit	Submit the profile form after setting the program field to workflowChangeAttributeRequest. The argument attr determines the attribute for the change attribute request and the argument action determines whether this is a remove or change request.

Table 25 Miscsc.js

SCRequiredValues Alert	Provide an alert to the user saying that the required number of values (specified by the argument num_req) for the field ob_name (another argument) is not met.
SCPasswordNo Confirm	Alert the user that the password values specified by the argument ob_name do not match.
SCCheckSingleForm	Perform validation for each field within the form and alert the user if the validation is not satisfied. Otherwise return true. Sample validation includes checking that password matches, number of required values are met, and so on.
SCCheckForm	Check all forms on the page to make sure they satisfy the validation requirements.
SCSwitchProfile	Switch between horizontal and vertical modify profile by switching stylesheets.
SCSubmit	Submit the form by first performing all the validation. For the different types of actions (for example, save, less_values), additional logic is applied, such as pop up messages displayed and additional information sent back to the COREid server.
getElementIndex	Returns the index of the element elementName in the form, or -1 if elementName is not found.
CopyElements	Assigns the value of elements in formFrom to the element of the same name in formTo.
hasKeyGenForm	Make sure the NavGenKeyform form is present.
InvokeKeyGen	A helper function that submits the NavGenKeyform.
AppendFormElements	Appends all the elements from all forms in all layers within the document to the input parameter form. form is in most cases the profileForm which is the form that is submitted. This method assigns the value of each element of each form within each layer to the identically named element in form. This method does not yet handle all special case display types; currently only checkbox is handled.

Monitorwf.js

Located in *WebPass_install_dir*\identity\oblix\lang\shared, Monitorwf.js includes the functions described in Table 26.

Table 26 Monitorwf.js

startSearchAnd Submit	Validate the monitorsearchform and then submit it.
continueSearch	Called when the user clicks the back and next buttons in monitor search results. If next is pressed, first checks if we are on the last record set, if so signals an error, otherwise, submits the form. If back is pressed, first checks if we are on the first set of records, if so signals that we are already at the beginning, otherwise, submits the form.
doSearchAndSort	Called to sort the search results for monitor search results using the arguments sortBy and sortOrder to perform the sorting.
submitresults	Called to purge/archive/unlock a workflow.
toggleselect	Toggle the search results form checkboxes from false to true.

Unspecified Program Names

For some screens, usually the most basic ones, the URL may not contain the program name. The program name can always be determined by generating and capturing the OutPutXML using format=XML. The stylesheet name will be included, at the top of the resulting file.

Behavior with the result will vary with different versions of Microsoft Internet Explorer. For consistency, use Netscape Communicator and save the output as a text file, using an xml extension. You can then view this file with any text editor to get the stylesheet name.

Customizing NetPoint

XSL stylesheets give you the ability to place almost any NetPoint component or piece of data almost anywhere on a page. This is more of an art than a science, and this Guide does *not* attempt to give precise instructions for getting the presentation you want. Instead, the following discussions outline the recommended approach for a minor change using the default Classic Style as a foundation.

Task overview: Customizing styles

1. Complete the “Prerequisites to Customizing Styles” on page 81
2. Review the “Customization Facts” on page 81.
3. Review the “Customization Guidelines” on page 83.
4. Familiarize yourself with the “Customization Methodology Checklist” on page 84.
5. Complete the task “Customizing the NetPoint GUI” on page 85 to gain first-hand experience with customization and testing to debug your process.
6. See also, “Modifying Catalog Files” on page 141.

Prerequisites to Customizing Styles

Be sure to complete the prerequisites below before you start to customize a style in NetPoint. This allows you to keep the original NetPoint Classic Style (\style0) intact for reference and in case you need to return to it as a last resort.

To prepare to customize styles in NetPoint

1. As an Identity Administrator, add your own style to NetPoint as described in the *NetPoint 7.0 Administration Guide*.

   The original NetPoint style stays in effect until an Identity Administrator makes your style the new system default.

2. As an Identity Administrator, select the new style as the default style in NetPoint so that you can see the effect of any changes you make.

Customization Facts

Style Updates and Maintenance—Default NetPoint wrapper files in \style0 and default global stylesheets in \shared are periodically updated by Oblix to instantiate improvements through patches and product upgrades.

The NetPoint Release Notes will notify you when such updates occur so you can propagate the changes to your custom styles. You will need to compare the new NetPoint file with your custom file and propagate any changes to your custom styles. It is risky to overwrite a default style with a customized style that bears the same name.

Be sure to record the changes you make and the files that are involved so you can more quickly update custom stylesheets when NetPoint updates default styles.

Custom Directory—Stylesheet customization should occur only within your custom directory. Customized stylesheets must reside in your custom directory and relative pointers in all files must point to the files in your custom directory, *not* to files in \shared.

Registration Files—As discussed in "General Content of Registration Files" on page 32, a common registration file and each application's registration file contain the names of the stylesheets and schema files needed to present pages for the application. For example, when you look at the User Manager registration file in identity\oblix\apps\userservcenter\userservcenterreg.xml, you can see the application name and the names of the stylesheets the application calls during the completion of various functions.

Also, given the application and the program name, you can locate the corresponding schema file name in the application's registration file.

Oblix recommends that only experienced developers using *extreme care* consider editing a registration file. Registration files are covered in more detail at "Registration Files" on page 31.

Pointers—All wrapper files and stylesheets contain pointers as include statements that call another file. Most of these pointers are relative pointers that indicate where within the directory structure the file is without providing an absolute path name.

For example, when you look at the usc_profile.xsl stylesheet called by User Manager functions, you can see that it contains include statements with relative pointers that call the following files:

./basic.xsl
./selectorinfo.xsl
./usc_searchform.xsl
./usc_navbar.xsl

When you change the location of a file (place a copy of a stylesheet in your custom directory for customization), pointers to this file (whether relative or absolute) must be changed to reflect the new location in every file that calls it. All relative pointers in a stylesheet should point to files in your custom directory.

In addition, many stylesheets contain relative pointers to object files. If NetPoint cannot instantiate an object when the page is loaded, unexpected behavior may result. All relative pointers to object files should be absolute pointers, as discussed in "Editing Stylesheets" on page 94.

Wrapper Files—Wrapper files include pointers to actual stylesheets in \shared. However, you cannot be assured that a wrapper file will be called before the stylesheet because both the common registration file and the application's own registration file call stylesheets according to an internal ordering. For this reason, all wrapper files in your custom directory must be overwritten by a copy of the corresponding default stylesheet from the \shared directory.

Important: Customizing stylesheets is an iterative process. Attempting to copy the entire contents of \shared into your custom directory at one time will produce an error.

Rather than copying all stylesheets at once, you start by investigating registration files to learn which functions (programs) call which stylesheets. You then selectively copy base stylesheets and a function-related stylesheet into your custom directory to overwrite their wrapper files, as discussed in “Copying Stylesheets to Your Custom Directory” on page 89. You then customize and test the style for that function. When this returns satisfactory results you repeat the process to customize another function.

Customization Guidelines

The guidelines below should help ensure a successful customization.

- Retain all original files in the \style0 and \shared directories in pristine condition and store them safely for future use. Also, make a backup copy of your customized style files so that patches won’t disrupt your customization.
- Record all changes you make and the files that are affected.
- Customize and test your new styles in a *non-production* environment before migrating them to your production environment.

Important: Oblix recommends that you do *not* modify original style files in the \shared or \style0 directories. These may be overwritten by patch updates and product upgrades or you may want to refer to them later.

- When you use only one style, consider breaking the dependence on stylesheets in the \shared directory (again, to prevent patch\release updates to \style0 and \shared from disrupting customizations). This means that no stylesheet in your custom directory should inherit from or reference a stylesheet in \shared or \style0.
- When you use multiple custom styles, consider the pros and cons of sharing customizations between multiple custom styles vs. implementing individual customizations per custom style. For example:

  a) **Two styles that share the same stylesheet**—When two custom styles (*custom_style1* and *custom_style2*) can *share* the same stylesheet you may be tempted to customize the stylesheet in the \shared directory despite the risk of having your custom style overwritten by an updated stylesheet in a product patch or upgrade.

  b) **Two individual styles**—When two custom styles (*custom_style1* and *custom_style2*) require their individually customized stylesheets you use the standard methodology and overwrite the wrapper files in your custom directory with the corresponding stylesheets in \shared.

- Consider using parameter stylesheet files for a custom style collection, rather than using hard-coded values (tab id’s, attribute names, table/link properties, and so on); this is similar to how program code is written using header files.

Customization Methodology Checklist

As mentioned earlier, customization is an iterative process and more of an art than a science. This Guide does *not* attempt to give precise instructions for getting the presentation you want. Instead, this section outlines the recommended approach for a minor change.

Note: Oblix recommends that you focus on stylesheets for one function at a time. Attempting to copy all stylesheets from \shared into your custom style directory will result in an error.

Table 27 Customization Methodology Checklist

Check	Action	Description
	Add a New Style	See the *NetPoint 7.0 Administration Guide* for details about adding a style to NetPoint and selecting your new style as the default.
	Choose a Function to Customize	Decide which function you will customize first. Oblix recommends that you customize stylesheets related to one function at a time.
	Copy Selected Stylesheets into Your Custom Directory	Copy selected stylesheets from \shared to your custom directory to overwrite corresponding wrapper stylesheets: • Base stylesheets • Stylesheets included in base stylesheets • A function-related stylesheet identified in application registration file • Function-related stylesheets identified in oblixbasereg.xml
	Customize Stylesheets in Your Custom Directory	• Change relative pointers in copied stylesheets to point to files in your custom directory. • Change relative pointers to objects to absolute pointers. • Complete other changes to implement the function's customization.
	Record Your Work	Keep a record of the files you change and the changes you make.
	Copy Your Custom Directory Structure to WebPass	Build a custom directory structure on WebPass and copy customized styles and images into it. **Note**: On WebPass, stylesheets are used only for client-side processing and are not required for server-side processing.
	Test Your Customized Style	• Test the customized style and make any alterations you need to the stylesheets in your custom directory. • Record the changes.
	Customize Another Function	Repeat the process above on a function by function basis: • Choose a function. • Copy related stylesheets from \shared to your custom directory. • Customize pointers and styles. • Record and test your work.

Table 27 Customization Methodology Checklist

Check	Action	Description
	Propagate the Customized Style	When you have copied and customized all stylesheets for the application, copy the custom style directory to all COREid Servers and WebPass hosts in your environment.

Customizing the NetPoint GUI

This example shows a method for changing the way a page looks, without changing what it does. The change is a simple font color alteration for a specific page in one application. After making the change you will verify that the change is successful. When you finish this functional customization, you will create the same custom style directory structure on WebPass and copy all image files into it so WebPass can display the appropriate images in response to queries. You then test the implementation.

The following topics demonstrate one sequence in the "Customization Methodology Checklist" on page 84. You may complete procedures below to gain first-hand experience:

- "Completing Prerequisites" on page 85
- "Choosing a Function to Customize" on page 88
- "Copying Stylesheets to Your Custom Directory" on page 89
- "Editing Stylesheets" on page 94
- "Copying Images and Styles to WebPass" on page 95
- "Testing Your Customized Style" on page 96
- "Propagating Styles" on page 97

For details about localizing messages, see "Localizing XSL Files" on page 98.

Completing Prerequisites

A prerequisite to customizing a style is to add a style to NetPoint and select the new style as the default, as described in the *NetPoint 7.0 Administration Guide*. The resulting files and file structure provide the foundation for your customization.

Suppose you added a new style named *Pastel* in a directory named *Pastel* and requested files be copied from Classic Style (in directory \style0).

To confirm the results of adding a new style to NetPoint

1. Add a style and select it as the default, as described in the *NetPoint 7.0 Administration Guide*.

New Custom Directory—NetPoint creates a directory that duplicates \style0 for the default language, English. If you have installed a Language Pack for French, NetPoint also creates a directory that duplicates \style0 in the French language directory.

2. Locate your new custom directory.

   For example:

   COREid_install_dir\identity\oblix\lang\en-us*Pastel*
   COREid_install_dir\identity\oblix\lang\fr-fr*Pastel*

   Wrapper Stylesheets—Your custom directory contains wrapper stylesheets that point to actual stylesheets in another directory. If you selected the Classic Style to copy from, your custom directory duplicates the content of the \style0 directory.

3. Open a wrapper stylesheet in your new custom directory, basic.xsl, and review the files that it includes.

   For this example:

   COREid_install_dir\identity\oblix\lang\en-us*Pastel*\basic.xsl

```
<?xml version="1.0" ?>
- <!--    Copyright (c) 1996-2001, Oblix Inc. All Rights Reserved.
  -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:oblix="http://www.oblix.com/">
  <xsl:include href="./style.xsl" />
  <xsl:include href="../msgctlg.xsl" />
  <xsl:include href="../../shared/basic.xsl" />
  </xsl:stylesheet>
```

   The basic.xsl wrapper stylesheet includes the three files below:

   - style.xsl file in your custom directory
   - msgctlg.xsl, one directory up from your custom directory (in identity\oblix\lang\en-us)
   - basic.xsl in identity\oblix\lang\shared

4. Locate and review the content of the basic.xsl stylesheet in \shared.

   For example:

   COREid_install_dir\identity\oblix\lang\shared\basic.xsl

```
<?xml version="1.0" ?>
- <!--   Copyright (c) 1996-2002, Oblix Inc. All Rights Reserved. -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:oblix="http://www.oblix.com/">
  <xsl:include href="obstringutil.xsl" />
```

```
- <!--  xsl:output indent="no"/ -->
  <xsl:include href="font.xsl" />
  <xsl:include href="title.xsl" />
...
```

The basic.xsl stylesheet in the \shared directory includes additional files (font.xsl, title.xsl, obstringutil.xsl) and provides templates to define attributes and status and control display information. See “basic.xsl” on page 48 for more information.

During your customization process, you will copy selected stylesheets from the \shared directory into your custom directory. This will overwrite wrapper files with corresponding stylesheets you can then edit in your custom directory.

New Custom XML Document—In addition to the custom directory structure, when you select the new custom style as the default style, NetPoint creates an XML document (a duplicate of style0.xml) named after the directory you created.

5. Locate and open the custom xml document that was created when you added the new style to NetPoint.

   For this example:

   COREid_install_dir\identity\oblix\config\style*Pastel*.xml

```
<?xml version="1.0" ?>
- <ParamsCtlg xmlns="http://www.oblix.com" CtlgName="style0">
- <ValNameList ListName="">
<NameValPair ParamName="styleReady" Value="TRUE" />
</ValNameList>
</ParamsCtlg>
```

   This new file, stored with style0.xml, provides the status of your custom style and the location of the original style directory from which wrapper files were copied. For example, if your custom style directory is named *Pastel* and you copied from Classic Style, the *Pastel*.xml file shown above is created when you select Pastel as the default style.

   You do not need to edit this file. The original style0.xml remains unchanged. Also, there is a .lck version, Pastel.xml.lck, which is a lock file. No other new files are created when you add a new style to NetPoint.

   Updated styles.xml—The styles.xml file is updated to include a new NameValPair that provides both the directory and style names you supplied when creating the style.

6. Locate and open the styles.xml file to confirm it was updated with your new style information.

   For example:

   COREid_install_dir\identity\oblix\config\style\styles.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <ValNameList xmlns="http://www.oblix.com"
ListName="styles.xml">
<NameValPair ParamName="style0" Value="Classic Style" />
<NameValPair ParamName="Pastel" Value="Pastel" />
</ValNameList>
```

   Notice, in the example above, that both the default Classic Style and new custom *Pastel* style are identified. You do *not* need to edit this file.

After confirming your custom directory structure, new and updated files, you are ready to choose a function and begin your customization.

Choosing a Function to Customize

The first step in the customization process is to choose a function to customize. For this example, suppose you want to change the font color to red on a specific page of the User Manager without changing anything else.

To identify the function and source information

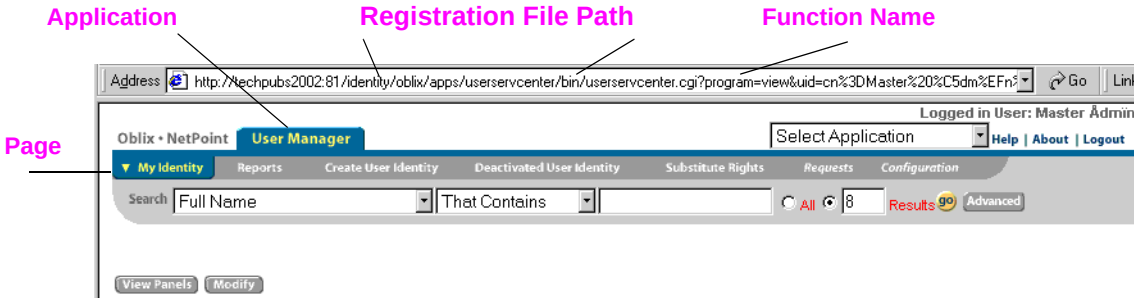
1. Log in to NetPoint, as usual.
2. Navigate to the desired page in NetPoint.

   For this example, click:

   COREid System Console > User Manager > My Identity

   The page appears, as shown in Figure 3.

Figure 2 Customization Example: A User Manager Page Displays Red Text



When you display a page in NetPoint, the following information is useful when you begin customizing styles. In Figure 3:

- **Application Name**—The application name, User Manager, appears on a highlighted tab in the top left area of the screen.

  Each application's \bin directory contains the registration file you need to identify functions. See "Registration Files" on page 31.

- **Page Name**—The page name, My Identity, is the first one you want to customize so you can see text in a red font color.

- **Registration File Path**—The URL for each page includes a path to the application page, identity\oblix\apps\userservcenter\bin\userservcenter.cgi in this case. You can use this to locate the relevant registration file on the COREid Server.

- **Function Name**—The URL for each page also includes a segment, *program=view* in this case, that you can use to locate the relevant stylesheet name for the function in the registration file.

3. Record the required information to assist you during the customization.

   For this example:

   Application—User Manager
   Page—My Identity
   Registration File Path—
   COREid_install_dir\identity\oblix\apps\userservcenter\bin\
   Function—program=view

Copying Stylesheets to Your Custom Directory

Once you have identified the function you want to customize, your next task in any customization is to copy relevant stylesheets into your custom directory from the \shared directory. This will overwrite wrapper files in your custom directory with copies of stylesheets you can customize. This also retains the original stylesheets in \shared as well as the original default wrappers in \style0.

Locating and copying relevant stylesheets is an iterative process in itself. In the following procedure you will locate and copy:

- Base stylesheets
- Stylesheets *included* in base stylesheets
- The specific function-related stylesheet identified for the program in the application's registration file, in this case the stylesheet associated with program=view
- Stylesheets *included* in the function-related stylesheet

Eventually your custom directory will contain all stylesheets, including those identified in the application's registration file and in oblixbasereg.xml. Even if you do not need to edit a stylesheet, it must be copied to your custom directory.

Important: Copying stylesheets is an iterative process that must be done in a selective manner. Attempting to copy all stylesheets from \shared to you custom directory at one time will result in an error.

To locate and copy relevant stylesheets

1. Copy the *base* stylesheets to your custom style directory from \shared to *overwrite* the default wrappers with stylesheets you can customize.

   For example:

   Copy from—*COREid_install_dir*\identity\oblix\lang\en-us\shared\ basic.xsl, font.xsl, searchform.xsl, navbar.xsl, title.xsl

   Copy to—*COREid_install_dir*\identity\oblix\lang\en-us*Pastel*\

   This retains the original base stylesheets in \shared as well as the original default wrappers in \style0.

2. Open each base stylesheet in your custom style directory and locate include statements that point to other stylesheets you need to copy, as well as any style information you need to customize.

   For this example, see Table 28:

Table 28 Base Stylesheet Pointers and Items to Customize

Base Stylesheets in Custom Directory	Pointers to Related Stylesheets and Items to Customize

basic.xsl	Contains implied relative include pointers to other stylesheets you need in your local custom directory:
	`<xsl:include href="obstringutil.xsl" />`
	`<xsl:include href="font.xsl" />`
	`<xsl:include href="title.xsl" />`
	
	Record the names of additional stylesheets you need to copy into your custom directory from \shared. In this case, obstringutil.xsl.

font.xsl	Does not contain include pointers to other files.
	Does contain color information you will customize:
	`<xsl:variable name="subHeadingColor">#006699...`
	`<xsl:variable name="contentTitleColor">#000000...`
	`<xsl:variable name="contentTextColor">#000000...`

searchform.xsl	Does not contain include pointers to other files. Does not contain color information you will customize. No changes needed to this stylesheet in your custom directory.
navbar.xsl	Does not contain include pointers to other files. Does contain color information you may customize later.

Table 28 Base Stylesheet Pointers and Items to Customize

Base Stylesheets in Custom Directory	Pointers to Related Stylesheets and Items to Customize
title.xsl	Does not contain include pointers to other files. Does contain color information you may customize later. No changes needed to this stylesheet in your custom directory.

3. Copy stylesheets *included* in base stylesheets to your custom directory from \shared.

   For this example, obstringutil.xsl:

   Copy from—*COREid_install_dir*\identity\oblix\lang\en-us\shared\obstringutil.xsl

   Copy to—*COREid_install_dir*\identity\oblix\lang\en-us*Pastel*\obstringutil.xsl

4. Record the stylesheets you have copied from \shared to your custom directory so you can track your work.

5. Locate the required registration files.

   For this example, oblixbasereg.xml and userservcenterreg.xml:

   COREid_install_dir\identity\oblix\apps\common\bin\oblixbasereg.xml
   COREid_install_dir\identity\oblix\apps\userservcenter\bin\userservcenterreg.xml

   At some point, you typically need stylesheets included in the common registration file oblixbasereg.xml. However, stylesheets included in oblixbasereg.xml are not needed for this example.

   For this example, you need to locate only the function-related stylesheet in the userservcenterreg.xml file.

6. Open the application's registration file and locate the function-related stylesheet you need.

   For this example, locate ObProgram name="view":

```
   <?xml version="1.0" ?>
- <ObProgramRegistry>
   - <ObApplication name="userservcenter">
      - <ObProgram name="front">
         <ObStyleSheet name="usc_profile.xsl" />
         <ObSchema name="usc_front.xsd" />
      </ObProgram>
      - <ObProgram name="commonNavbar">
         <ObStyleSheet name="usc_profile.xsl" />
         <ObSchema name="usc_front.xsd" />
      </ObProgram>
```

```
...
   - <ObProgram name="view">
      <ObStyleSheet name="usc_profile.xsl" />
      <ObButton name="initiateDeactivateUser" />
      - <!-- ObButton name="manageSubscriptions"/ -->
      <ObButton name="userreactivate" />
      <ObButton name="wfTicketDelete" />
      <ObButton name="userModify" />
      <ObSchema name="usc_profile.xsd" />
   </ObProgram>
...
```

You can see in the registration file that the usc_profile.xsl stylesheet is associated with the function you want to customize (ObProgram name="view"). The usc_profile.xsl stylesheet is also associated with a number of other functions.

7. Copy the function-related stylesheet, usc_profile.xsl, to your custom style directory from \shared and record the stylesheet name.

   For this example:

   Copy From—
   COREid_install_dir\identity\oblix\lang\en-us\shared\usc_profile.xsl
   Copy To—
   COREid_install_dir\identity\oblix\lang\en-us*Pastel*\usc_profile.xsl

8. Open the function-related stylesheet and locate include statements that point to other stylesheets you need to copy, record any information you need to customize.

   For this example, usc_profile.xsl:

Table 29 usc_profile.xsl Pointers and Items to Customize

usc_profile.xsl in Custom Directory	Pointers to Related Stylesheets and Items to Customize

usc_profile.xsl	This main stylesheet for the User Manager includes stylesheets that need to be copied to your custom directory:
	
	`<xsl:include href="./basic.xsl" />`
	`<xsl:include href="./selectorinfo.xsl" />`
	`<xsl:include href="./usc_searchform.xsl" />`
	`<xsl:include href="./usc_navbar.xsl" />`
	
	Note: selectorinfo.xsl, usc_searchform.xsl and usc_navbar.xsl should be copied.
	
	Also record pointers to objects that should be customized:
	`<object id="cenroll" classid= ...`
	`codebase="../../common/bin/xenroll.cab" />`
	and
	`<script src="../../common/bin/installCert.vbx" ...`

9. Repeat steps to copy relevant stylesheets, then record their names and details you need to change.

   For this example:

   Copy From—

   COREid_install_dir\identity\oblix\lang\en-us\shared\selectorinfo.xsl
   COREid_install_dir\identity\oblix\lang\en-us\shared\usc_searchform.xsl
   COREid_install_dir\identity\oblix\lang\en-us\shared\usc_navbar.xsl

   Copy To—

   COREid_install_dir\identity\oblix\lang\en-us\Pastel\selectorinfo.xsl
   COREid_install_dir\identity\oblix\lang\en-us\Pastel\usc_searchform.xsl
   COREid_install_dir\identity\oblix\lang\en-us\Pastel\usc_navbar.xsl

   These stylesheets do not contain include statements, other stylesheet names, nor parameters you need to change.

   You have collected, copied, and recorded relevant stylesheets for this example.

Editing Stylesheets

After copying relevant stylesheets, you may need to edit them. As described in Table 28, the information that needs to be customized for this example includes:

- Font colors defined in the base stylesheet font.xsl should be changed to red.
- Pointers to objects defined in usc_profile.xsl should change from a relative path to an absolute path.

Note: To help streamline development and testing, consider implementing XSL stylesheet control parameters. See “Caching Considerations” on page 24.

To edit stylesheets for a simple font color change

1. Open the font.xsl stylesheet in your custom directory in a text editor.

   For example,

   COREid_install_dir\identity\oblix\lang\en-us*Pastel*\font.xsl

2. Edit the stylesheet to change all colors from the default color to red (FF0000), then save the change.

   For example,

   Change all Default Font Colors From—

```
... <xsl:variable name="pageHeaderColor">#006699</xsl:variable>
<xsl:variable name="subHeadingColor">#006699</xsl:variable>
<xsl:variable name="contentTitleColor">#000000</xsl:variable>
<xsl:variable name="contentTextColor">#000000</xsl:variable>
and others ...
```

   To Red (#FF0000)—

```
... <xsl:variable name="pageHeaderColor">#FF0000</xsl:variable>
<xsl:variable name="subHeadingColor">#FF0000</xsl:variable>
<xsl:variable name="contentTitleColor">#FF0000</xsl:variable>
<xsl:variable name="contentTextColor">#FF0000</xsl:variable>
and others ...
```

3. Record your changes to this file.

   If you restarted the COREid Server now you would not yet see your changes. This is because you have not yet customized the function-related stylesheet that identifies where to apply the changes.

4. Edit the basic.xsl stylesheet in your custom directory as directed below to add required include statements that were in the original basic.xml (but were lost when you copied over the basic.xsl from the shared folder).

   a) Locate the line containing the following:

   <xsl:include href="obstringutil.xsl"/>

b) Add the information below *above* the line identified in a):

<xsl:include href="./style.xsl" />
<xsl:include href="../msgctlg.xsl" />

5. Edit the usc_profile.xsl stylesheet in your custom directory to change the relative path to objects, as indicated below, then save the changes.

   For example:

   Change From a Relative Path—

```
- <head>
...<object id="cenroll" classid="clsid:43F8F289-7A20-11D0-8F06-00C04FC295E1"
codebase="../../common/bin/xenroll.cab" />
... <script src="../../common/bin/installCert.vbx" language="VBScript" />
</head>
```

   Change To an Absolute Path—

```
- <head>
... <object id="cenroll" classid="clsid:43F8F289-7A20-11D0-8F06-00C04FC295E1"
codebase="/identity/oblix/apps/common/bin/xenroll.cab" />
... <script src="/identity/oblix/apps/common/bin/installCert.vbx"
language="VBScript" />
</head>
```

   This concludes the specific function-related change for this example.

6. Ensure that filesystem access control for new custom style directories and files is set to match the ownership and permissions of \style0.

7. Restart the COREid Server.

   If you log in to NetPoint now and view the My Identity page, you will see the red font color. However, the images supplied by WebPass won't appear until they are included in a corresponding custom style directory structure on the WebPass host.

Copying Images and Styles to WebPass

Images and JavaScript are served by the Web server that the WebPass is installed against, not by the COREid Server. When a style refers to an image, the image is served by WebPass. If the image does not exist in the WebPass file hierarchy, the image will appear as a broken link. To avoid this, you need to create a custom style directory on WebPass and include all images in this structure, whether you are adding new images or using default images.

To copy images to WebPass

1. Copy your custom style directory from the COREid Server to WebPass.

   For example:

Copy From—*COREid_install_dir*\identity\oblix\lang\en-us*Pastel*
Copy To—*WebPass_install_dir*\identity\oblix\lang\en-us*Pastel*

Note: Stylesheets are included on WebPass for use with client-side processing only. Stylesheets are not required on WebPass for server-side processing.

---

2. Copy all image files from \style0 on WebPass to your custom directory on WebPass, whether you are using default images or adding new images.

   For example:

   Copy images From—*WebPass_install_dir*\identity\oblix\lang\en-us\style0
   Copy images To—*WebPass_install_dir*\identity\oblix\lang\en-us*Pastel*

   Note: The example above does not include new images; only default images called by the new custom style. If custom images are included, copy those to the custom directory as well.

   ---

3. Restart WebPass.

You are ready to test your customized style.

Testing Your Customized Style

You are ready to test your customized style and make any changes needed to achieve satisfactory results.

Note: To help simplify development and testing, you may want to implement XSL stylesheet control parameters, as discussed in “Caching Considerations” on page 24. You may use an XML editing environment to allow testing stylesheet customizations offline, as discussed in “Useful Tools” on page 183.

If you don’t obtain the desired result, check the items in “Troubleshooting Customization Issues” on page 98.

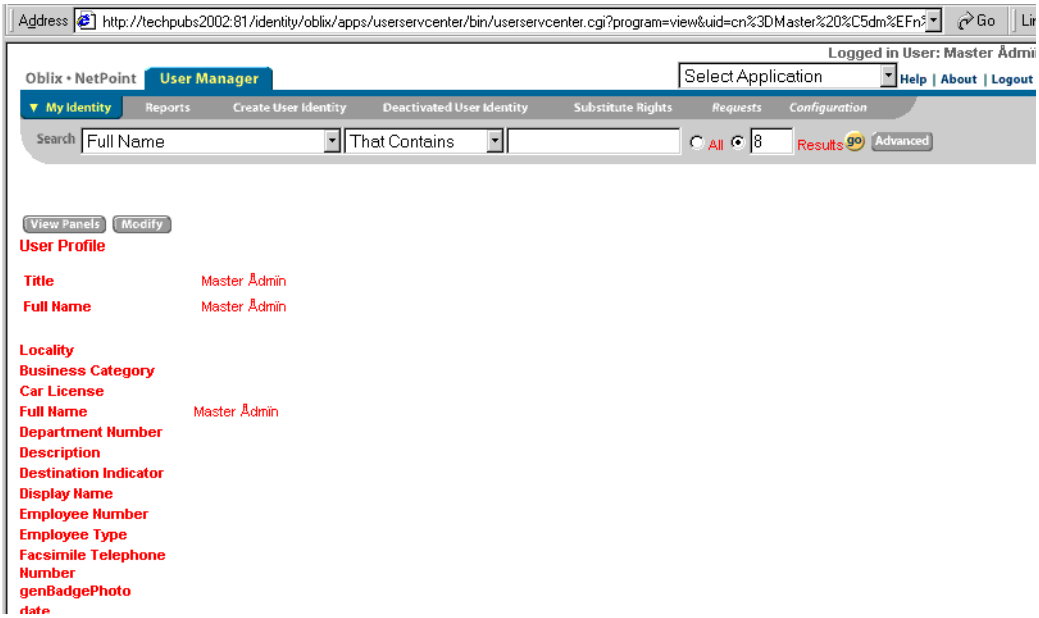
To test your style

1. Log in to the COREid System, as usual.
2. Navigate to the page you customized.

   For example:

   User Manager > My Identity
3. Confirm that the font color now displays in red, as shown in Figure 3.

Figure 3 Customization Example: A User Manager Page Displays Red Text



Propagating Styles

When you know your style is working, you can push this out to other NetPoint systems.

To propagate styles

1. Copy your custom style directory from the COREid Server to all other COREid Servers.
2. Restart each COREid Server.
3. Copy your custom style directory from the WebPass host to all other WebPass hosts.
4. Restart each WebPass.

Troubleshooting Customization Issues

If you obtain unexpected results, check the items below to ensure that you have completed all tasks correctly.

- Have you added a new style to NetPoint and selected this as the default style?
- Have you identified and copied relevant stylesheets to your custom directory from \shared (see application and common registration files):
  - Base stylesheets
  - Stylesheets *included* in base stylesheets
  - The specific function-related stylesheet identified in the application's registration file.
  - Stylesheets *included* in the function-related stylesheet?
  - Relevant stylesheets in the common registration file?
- Have you made appropriate changes to stylesheets in your custom style directory?
  - Relative pointers to stylesheets
  - Relative pointers that should be absolute pointers to objects
  - Other customization details
- Have you created a duplicate custom style directory structure on WebPass?
- Have you copied images to your custom style directory structure on WebPass?

  Note: NetPoint relies on these images and Oblix recommends that you copy the files to your custom directory. On rare occasions, if your custom image does appear but the default does, you may need to change the default image.

- Have you restarted the COREid Server and WebPass?

For more information, see "Troubleshooting Example" on page 191.

Localizing XSL Files

As discussed elsewhere, multiple languages are available for use with NetPoint 7.0. Messages that were once in stylesheets are language dependent and are now defined separately as variables in message catalogs. The new NetPoint directory structure consolidates all message catalogs for JavaScript files, XSL, and HTML.

- Any language-specific files will be located in \lang*langTag*.
- Any non-language specific objects are located within \lang\shared.

All the stylesheets have a language-specific wrapper in \lang*langTag*\style0 which *includes* the main language-neutral version stylesheet in \lang\shared. This new wrapper segregates the main stylesheet functionality, which is language independent, from language-specific messages.

Language-specific messages are referred to through variables in message catalog files, as discussed in:

- “Handling Language-Specific Stylesheet Messages” on page 149
- “Handling Language-Specific Messages for JavaScript” on page 150

Display names for the COREid applications are stored in the stylesheet. For localization, these display names must be translated for the each language that your organization supports.

For example, the display name User Profile is stored as an XSL variable “MUserProfile” in:

COREid_install_dir\identity\oblix\lang\en-us\msgctlg.xsl file

The XSL variables are stored in the XSL stylesheet, for example usc_profile.xsl, located in *COREid_install_dir*/identity/oblix/lang/shared.

To localize XSL files

1. Store the display name text strings in a separate file as XSL variables.
2. Reference them in the stylesheet.

3 Customizing Portal Inserts

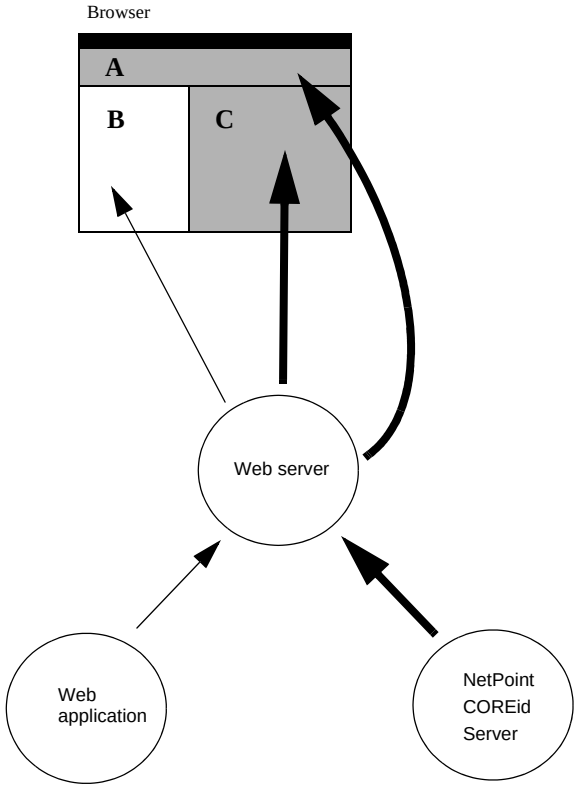
NetPoint Portal Inserts provide a way to insert content generated by NetPoint into other applications without programming. The typical use of this is to build a subset of NetPoint functionality into your own Web application. You can, for instance, use NetPoint COREid System's searching capabilities to add a company directory search feature to your site.

This chapter describes how portal inserts work and how to implement them with your NetPoint installation:

- "Overview of Portal Inserts" on page 102 provides a short description of Portal Inserts.
- "Using Portal Inserts" on page 103 describes the generic method for using Portal Inserts, including the URL format to be used.
- "Portal ID/BackURL" on page 106 talks about a method that provides a quick return to the calling portal.
- "COREid Applications and Portal Inserts" on page 108 locates the specific COREid applications that use Portal Inserts and lists the functions that each one supports.
- "Parameter Reference" on page 121 lists and describes the parameters used by the functions.
- "Portal Inserts Example" on page 132 gives an example for using Portal Inserts.

Overview of Portal Inserts

The diagram below illustrates how you might construct a Web application with the help of NetPoint Portal Inserts:



In this scenario, the Web application uses three HTML frames (A, B and C) to lay out its content. Frame A might provide a search function, the search controls themselves being provided by a NetPoint Portal Insert. Frame B might contain an unrelated report using non-identity information, generated by the application itself.

Frame C might contain detailed information about a particular identity, such as you see in User Manager's profile page. The contents of this frame could be provided entirely by a second NetPoint Portal Insert requested from the search function in frame A. The Portal Insert may optionally be combined with a custom stylesheet to display the page in a way that better suits the Web application's look and feel. This kind of customization is described in "Designing the GUI with PresentationXML" on page 15.

Note: The application may also use NetPoint to generate the data contained in frame B, through the IdentityXML interface for instance. That page may contain links that can be used to request a new Portal Insert (with or without a custom stylesheet) for frame C.

Using Portal Inserts

To use Portal Inserts, you typically begin by identifying a NetPoint feature that you want to integrate into your application. Before accessing a function as a Portal Insert, it is a good idea to verify that you can access it as a NetPoint user.

When you have constructed the URL that generates the content you require, you add the request to your application wherever you want the content displayed. For instance, in a Web application, you might specify the URL as the target of a link, or as the source document for a frame.

NetPoint WebPass receives the HTTP request when the Portal Insert page is to be displayed. WebPass interprets the URL with its additional parameters as a portal request. If the URL contains invalid data, access is denied. Otherwise, an HTTP response is returned, typically providing much the same interactive HTML GUI as the base NetPoint system provides.

The URL format for Portal Inserts conforms to Internet RFC 2396 - *Uniform Resource Identifiers (URI): Generic Syntax*. The text of the RFC is located at:

```
http://www.ietf.org/rfc/rfc2396.txt
```

Specifically, the URL for a Portal Insert looks like this:

```
http://host:port/appname.cgi?
     param1=value1&param2=value2...
```

Note the following components

1. The NetPoint COREid Server location.

   This is the http://host:port/ part of the entry. It is exactly the same location you would use to enter NetPoint as a user.

   Note: The *http* scheme may be *https* if a secure connection is used.

2. The application location.

   This is the appname.cgi part of the entry. You point to the exact application, such as User Manager, that you want to use. A list of locations for each application is provided in “COREid Applications and Portal Inserts” on page 108. These are the “portals” to the functions that you want to carry out.

3. One or more sets of parameter name and value pairs.

   These are provided in the form param=value. The first set immediately follows the application choice and starts with a ?. The second and any additional sets start with a &. The parameters can be provided in any order.

   Remember that this text is being received as a URL and any non-URI characters, such as spaces and punctuation, that appear in parameter values must be encoded as discussed in the RFC.

   The following are some common characters and their URI-safe encoded equivalents:

Character Name	Character	URI-encoded Equivalent
space		%20
exclamation mark	!	%21
apostrophe	‘	%27
open parentheses	(	%28
close parentheses	)	%29
comma	,	%2C
colon	:	%3A
equals	=	%3D

The following is an example URL to access a portal insert, first as one long string and then with the parts broken out for discussion:

```
http://customer.com:81/identity/oblix/apps/userservcenter/
bin/userservcenter.cgi?
program=search&tab_id=employees&comp=true&
STy1=cn&SLk1=OSM&SSt1=john
```

The following is the meaning of each part:

- **http://customer.com:81/**—The server location.
- **identity/oblix/apps/userservcenter/bin/userservcenter.cgi**—The User Manager application location.
- **?program=search**—The first parameter, which in this case requests a search. The program parameter is always required. A good convention is to have it always be the first parameter. You can then quickly identify what function the request is expected to perform.

  Note: The leading `?` separates the resource from the parameter list. Only the first parameter starts with `?`. If there are subsequent parameters, they are delimited by the `&` character.

- **&tab_id=employees**—A second parameter, in this case requesting that the search be done under the employees tab.
- **&comp=true**—A third parameter, in this case comp, which specifies that only the requested information (and not the NetPoint navigation controls) is to be displayed.
- **&STy1=cn**—The first (indicated by the 1) search criterion is that the search is against the cn attribute.

  Note: You may specify further search criteria by passing more than one group of (STYn, SLkn, SStn) parameters. The n is always a number, used to group parameters together that belong to the same search criterion. See “Parameter Reference” on page 121 for more details, in particular the description of the noOfFields parameter.

- **&SLk1=OSM**—The search type is to be a substring match.
- **&SSt1=john**—The data containing the string john.

Portal ID/BackURL

Portal inserts provide a way to embed NetPoint functionality into user Web applications. Once the goal is accomplished, the user may be several layers deep in NetPoint screens. The user could return to the calling screen by using the browser back button several times. Another way is to use the Portal ID feature. This inserts a back button to the NetPoint screens. Clicking on this button returns the user to the calling portal or to any other user specified URL.

To use this feature you append the portalid parameter to the URL, provided at the calling portal, and specify a label for the return URL. The portalid parameter value persists, meaning that its value is known to all successive screens, all of which will contain the appropriate back button. The example URL provided earlier is easily modified to use the portalid parameter:

```
http://customer.com:81/identity/oblix/apps/userservcenter/
bin/userservcenter.cgi?
program=search&tab_id=employees&comp=true&
STy1=cn&SLk1=OSM&SSt1=john&portalid=mychoice
```

In this example, mychoice is a label for the precise URL that the portal designer wants to return to. The portal designer associates the label with the actual URL by changing the content of the Portal Inserts Caller Identification Parameter File (PICI Parameter File), portalidparams.xml. A generic version of this file is provided as part of the COREid installation, at

```
identity/oblix/apps/common/bin/portalidparams.xml
```

The installed content for this file is the following:

```
<?xml version="1.0"?>
<ParamsCtlg xmlns="http://www.oblix.com"
        CtlgName="portalidparams">
<CompoundList ListName="">
   <ValNameList ListName="oblix1" >
        <NameValPair ParamName="portalIdBackUrl"
             Value="http://www.oblix.com"/>
        <NameValPair ParamName="portalIdBackButton"
             Value="../../common/ui/style0/
             NAVportalreturn1.gif"/>
        <NameValPair ParamName=
             "portalIdBackButtonMouseOver"
             Value="Click here to go to oblix mainpage..1"/>
   </ValNameList>
   <ValNameList ListName="oblix2" >
        <NameValPair ParamName="portalIdBackUrl"
             Value="http://www.oblix.com"/>
        <NameValPair ParamName="portalIdBackButton"
             Value="../../common/ui/style0/
```

```
                    NAVportalreturn2.gif"/>
          <NameValPair ParamName=
               "portalIdBackButtonMouseOver"
               Value="Click here to go to oblix main

               page..2"/>
   </ValNameList>
</CompoundList>
</ParamsCtlg>
```

Note: The example contains two `portalid`s, `oblix1` and `oblix2`. You may add more.

The information provided for each `ValNameList` item in the file associates a user-created label with the return URL, the image of a back button, and mouseover text to be associated with the button. All four of these items can be changed by the user, as follows:

Parameter	Description
ListName	A unique id for the calling portal. This is any user defined label, mychoice in the earlier example. The id value none is reserved; it has special meaning to COREid. (See the discussion of the portalid parameter on page 125.).
Value for portalIDBackURL	A back URL. This could be the URL of the calling portal, or any other user specified URL.
Value for portalIDBackButton	The file path to the image to display for the back button. The button image is presented at the top of the page. When the user clicks on this button, the browser will return to the location specified in the value for the portalIDBackURL. The path can be relative to the *COREid_install_dir*/identity/oblix/apps/bin directory, or a fully specified URL.
Value for portalIDBackButton MouseOver	The mouse-over message for this button, that is displayed when the user puts the mouse cursor over the button.

The PICI Parameter file is loaded when COREid starts. If its content is subsequently changed, the user should reload the file in order to make the changes useable. Rather than stopping and starting COREid, you reload the file by entering the following URL to your browser:

```
http://host:port/identity/oblix/apps/admin/bin/
genconfig.cgi?program=flushCache&cachetype=portalid
```

COREid Applications and Portal Inserts

This section lists the applications that respond to portal requests. For each application, the text shown replaces the `appname.cgi` information in the URL format.

For the Group Manager application:

```
identity/oblix/apps/groupservcenter/bin/
groupservcenter.cgi
```

For Lost Password Management:

```
identity/oblix/apps/lost_pwd_mgmt/bin/lost_pwd_mgmt.cgi
```

For the Organization Manager application:

```
identity/oblix/apps/objservcenter/bin/objservcenter.cgi
```

For the User Manager application:

```
identity/oblix/apps/userservcenter/bin/userservcenter.cgi
```

Portal Insert Services

Each application provides one or more *functions* that can be accessed by URL parameters. These functions are also referred to as *services*. The functions fall into three major categories:

- **Present**—These services present standard NetPoint pages, for user interaction.
- **Get**—These services show current directory content, but do not change it.
- **Set**—These services change current directory content or perform an action, such as logging out.

All available functions are listed in the sections that follow, grouped under the above three categories. For each function, the following are provided:

- **Name**—This is the name of the `program` that carries out the function. It is good practice to use this parameter first, in the form `program=xxxx`, before appending other parameters.
- **Description**—An explanation of the function
- **Works with**—A list of applications with which the function works. Note that many of the functions work with more than one application.
- **Parameters**—A table of parameters that either *must* (REQ) or *may* (OPT) be used in a URL that invokes the function. If you fail to specify a required parameter in the URL, an error page is returned. If you specify any parameter name but no value, or an invalid value, an error page is returned.

The parameters themselves are described in detail starting at “Parameter Reference” on page 121.

In the function descriptions below, required parameters are listed first, followed by optional parameters. In the URL, you can specify parameters in any order you prefer, but because these URLs can become unwieldy, it is a good idea to follow a guideline such as *function*, followed by *required parameters*, followed by *optional parameters*.

The *Note* column calls attention to any issues that you should keep in mind while using the parameter with a particular function, but that are not part of the parameter’s description *per se*.

Note: You must have the appropriate rights assigned to you in order to use a function. Your searchbase must include the information you want to view or change. You require *read* rights for any attributes you expect to view, or tabs whose configured attributes you expect to view. You require *write* rights for any information you expect to change.

Functions to Present Pages

Following is a list of functions that present interactive pages.

delete

Description:Use this function to generate a page, including a delete button, from which you can delete a group or an organization. See workflowDeactivateUser to generate a page from which you can remove a *user*.

Works with:Group Manager, Organization Manager

Parameter	REQ/OPT
uid	REQ
comp	OPT

modify

Description: Use this function to present an interactive page from which data can be changed.

Works with: Group Manager, Organization Manager, User Manager

Parameter	REQ/OPT
uid	REQ

Parameter	REQ/OPT
comp	OPT

modifyLocation

Description: Use this function to display a page from which you can change the location of an individual or organization.

Works with: Organization Manager, User Manager

Parameter	REQ/OPT
locId	REQ
uid	REQ
comp	OPT
locObjClass	OPT
rectangle	OPT
scopeResolved	OPT
tab_id	OPT

passwordChallengeResponse

Description: Two URLs have been configured for your system, one to be used for password changes and the other for challenge response. This function sends the user to the challenge-response page, and from there, if the response is correct, to the password change page. From there, the function sends the user to the page specified by the backUrl.

Works with: Lost Password Management

Parameter	REQ/OPT
login	REQ
backUrl	OPT
target	OPT

predefinedReports

Description: Use this function to present an interactive page showing a set of predefined reports.

Works with: Group Manager, Organization Manager, User Managerr

Parameter	REQ/OPT	Note
comp	OPT	
tab_id	OPT	Default varies by application.

proxyAdmin

Description: Use this function to present an interactive page from which proxy administration can be done.

Works with: User Manager

Parameter	REQ/OPT
comp	OPT

redirectforchangepwd

Description:A URL will have been configured for your system, to be used for password changes. This function sends the user to the password change page. From there, the function sends the user to the page specified by the backUrl.

Works with: Lost Password Management

Parameter	REQ/OPT
login	REQ
backUrl	OPT
target	OPT

searchPage

Description: Use this function to present an interactive search page, where you can enter search parameters.

Works with: Group Manager, Organization Manager and User Manager.

Parameter	REQ/OPT	Note
advSearch	OPT	
comp	OPT	
tab_id	OPT	Default varies by application.

subscribe

Description: Use this function to present an interactive page from which you can subscribe to a group.

Works with: Group Manager

Parameter	REQ/OPT
uid	REQ
comp	OPT

viewLocations

Description: Use this function to get a page from which you can view the location of an organization or user.

Works with: Organization Manager, User Manager

Parameter	REQ/OPT
locId	REQ
uid	REQ
comp	OPT
coords	OPT
locObjClass	OPT
rectangle	OPT
scopeResolved	OPT
show_all	OPT
tab_id	OPT

workflowCreateProfile

Description: Use this function to present an interactive page from which a new entry can be created using a workflow.

Works with: Group Manager, Organization Manager, User Manager

Parameter	REQ/OPT	Note
comp	OPT	

Parameter	REQ/OPT	Note
tab_id	OPT	Default varies by application.

workflowDeactivateUser

Description: Use this function to present a page from which you can deactivate a user.

Works with: User Manager

Parameter	REQ/OPT	Note
uid	REQ	DN of the user who is to be deactivated.
ObWorkflowName	REQ	
comp	OPT	

workflowSelfRegistration

Description: Use this function to present a page from which you can add yourself to an organization, or as a user.

Works with: Organization Manager, User Manager

Parameter	REQ/OPT	Note
ObDomainName	REQ	
ObWorkflowName	REQ	
comp	OPT	
ObWfComment	OPT	
tab_id	OPT	Default varies by application.

workflowTicketSearchForm

Description: Use this function to present an interactive search page, where you can enter search parameters for specific tickets.

Works with: Group Manager, Organization Manager, User Manager

Parameter	REQ/OPT	Note
requestType	REQ	

Parameter	REQ/OPT	Note
comp	OPT	

unsubscribe

Description: Use this function to present an interactive page from which you can unsubscribe from a group.

Works with: Group Manager

Parameter	REQ/OPT
uid	REQ
comp	OPT

Functions to Get Data

Following is a list of all those functions that return data.

myGroupsProfile

Description: Use this function to get the profiles for groups you are a member, owner, or administrator of.

Works with: Group Manager

Parameter	REQ/OPT	Note
attrName	OPT	
comp	OPT	
showAdministrator OfGroups	OPT	At least one of the parameters names starting with show must be used.
showDynamic Groups	OPT	
showMemberOf Groups	OPT	
showNested Groups	OPT	
showOwnerOf Groups	OPT	

Parameter	REQ/OPT	Note
showStatic Groups	OPT	

search

Description: Use this function to present the result of a search.

Works with: Group Manager, Organization Manager, and User Manager.

Parameter	REQ/OPT	Note
SLk*n*	REQ	
SSt*n*	REQ	
STy*n*	REQ	
comp	OPT	
displayFormat	OPT	
noOfFields	OPT	
noOfRecords	OPT	
showAllResults	OPT	
sortBy	OPT	
sortOrder	OPT	
startFrom	OPT	
tab_id	OPT	Default varies by application.

showReportsResults

Description: Use this function to present the result of running a predefined report.

Works with: Group Manager, Organization Manager, User Manager

Parameter	REQ/OPT	Note
panel_id	REQ	
reportsubtab	REQ	Always takes the value predefinedReport.
reportName	REQ	

Parameter	REQ/OPT	Note
comp	OPT	
displayFormat	OPT	
noOfRecords	OPT	
showAllResults	OPT	
sortBy	OPT	
sortOrder	OPT	
tab_id	OPT	Default varies by application.

view

Description: Use this function to view selected attributes for a group, organization, or user.

Works with: Group Manager, Organization Manager, and User Manager

Parameter	REQ/OPT	Note
uid	REQ	DN of the user, group or organization whose attributes are to be viewed, depending upon the application. For User Manager only, this is optional. If no uid is specified, the profile of the logged in user will be shown.
attrName	OPT	If you do not use this parameter, then all attributes for the uid, that you are authorized to see, are returned. To get values for more than one attribute, use this parameter multiple times, once for each named attribute.
comp	OPT	

GroupMembers

Description: Use this function to view the members of a group.

Works with: Group Manager

Rights: Read rights on the *Member* attribute. Also, for dynamic members the read right on the *Dynamic Filter* attribute.

Parameter	REQ/OPT	Note
uid	REQ	DN of the group whose members are to be listed.
attrName	OPT	
comp	OPT	
showDynamicUserMembers	OPT	At least one of the show parameters in the list must be used, set to true.
showNestedUser Members	OPT	
showStaticUser Members	OPT	
SLk1	OPT	At most one set of these parameters is allowed with this function. The set is required if groupMemberSearch StringMinimumLength is not zero. See the parameter file “groupservcenterparams.xml” on page 202.
SSt1	OPT	
STy1	OPT	

workflowTicketInfo

Description: Use this function to get information about a specific request.

Works with: Group Manager, Organization Manager, User Manager

Parameter	REQ/OPT
workflowInstanceDn	REQ
workflowStepInstanceId	REQ
comp	OPT

workflowTicketSearch

Description: Use this function to present the result of a search for pending, completed, or all workflow requests.

Works with: Group Manager, Organization Manager, User Manager

Parameter	REQ/OPT	Note
requestType	REQ	If the required type is an outgoing request, then requestType is not needed.
targetApplication	REQ	
ticketType	REQ	If the required type is an outgoing request, then ticketType is not needed. If the required type is an incoming request, There are three possible entries. **WfAllTickets**—Search for all requests, regardless of status. **WfCompletedTickets**—Search for requests that have been completely processed. **WfPendingTickets**—Search for requests that are pending, only partially processed.
comp	OPT	
days	OPT	
noOfRecords	OPT	

Parameter	REQ/OPT	Note
sortBy	OPT	For workflow tickets, the class sorting attribute can have only one of the following values: obticketid (for Ticket Number) obapp (for Application Name) obactionname (forAction) obwfstatus (for Status) obwftypename (for Request Type) obtargetdn (for Requested For) obcurrentdn (for Requested by) obactordn (for Action Taker) obdateprocessed (for Date Processed) oblockedby (for Locked By) obsubflow (for Subflow Number) If the attribute is invalid, then an error message is returned, such as "Invalid value for parameter sortBy". If no attribute is specified, the default is the first attribute (most likely obticketid) in the admin-configured workflow ticket search table. (You can see this table by looking at the successive screens COREId system console > Common configuration >Configure workflow panels > Ticket search table).
sortOrder	OPT	
startFrom	OPT	

Functions to Set Data

Following is a list of all those functions that set data.

commonLogout

Description: Log out of Group Manager, Organization Manager, User Manager.

Works with: Group Manager, Organization Manager, User Manager.

Takes no parameters.

expandGroup

Description: Use this function to expand a dynamic group into its current static members.

Works with: Group Manager

Rights:VIEW for the *Group Dynamic Filter* and *Group Expansion* attributes; VIEW for the group class attribute; MODIFY for the *Member* attribute.

Parameter	REQ/OPT	Note
comp	OPT	
groupsToExpand	OPT	One or the other of these must be provided.
expandAllGroups	OPT	

workflowChangeAttributeRequest

Description: Use this function to initiate a change attribute request using a workflow.

Works with: Group Manager, Organization Manager, User Manager

Parameter	REQ/OPT
changeRequest Attr	REQ
changeRequest Type	REQ
ObWorkflowName	REQ
uid	REQ
comp	OPT

Parameter Reference

The following table describes each of the parameters in detail. In general, parameters have the same meaning for each function. When a function behaves differently with respect to a parameter, this is noted in the Notes column of the functions tables above.

Several parameters, such as `locId` and `tab_id,` take values that are not obvious from the GUI presentation. However, these can be obtained from the URL address for the function you want to implement as a portal.

For example, if you do a search within the User Manager and click on one of the employees to do a view, you see that the `tab_id` used is `employees`.

Boolean parameter values must be set to `true` or `false` in the URL. Alternative representations of Boolean values, such as yes and no, 1 and 0, etc. are not supported.

Integer parameter values must be specified as an uninterrupted sequence of decimal digits.

String parameter values must be URI-encoded as described above (see “Using Portal Inserts” on page 103) if they contain spaces or punctuation.

The values to be entered for many of the parameters listed here are the exact DN values as they appear in the directory, rather than the display values. To find these DN values, you will need to use a tool that will allow you to browse in the directory and display DN entries. An example of such a tool is `ldp.exe,` provided with Windows 2000 systems. Other methods are described below.

Find schema names for an attribute for an application by following these steps (taking User Manager as an example): COREid System Configuration > User Manager > User Manager Configuration > Configure Tab. Click on the link of the type of User you need, then click on **Modify Attributes**. At this point, an applet will show up. The top left corner shows a list of schema names for the attribute, and the top right corner shows the display names of the attributes, where you can locate the attribute you want to refer to.

Find attribute names by using the Modify Attributes feature for the appropriate application. For example, look under User Manager > User Manager Configuration > Configure Tab. Select the appropriate tab (which you are *not* going to change). Select **Modify Attributes**. The attribute names for that tab are displayed in the field identified as Attribute.

Parameters used for Portal Functions

Parameter Name	Description	Rules
advSearch	Use this parameter to specify that the advanced search form is to be used instead of the basic search form.	Single-valued, Boolean, true or false. Default: false.
attrName	Use this parameter to specify the names of one or more attributes to be viewed or changed, depending upon the function. Use the schema names, not the display names.	Multi-valued, string. Default: If no names are provided, then the attributes that will be shown are all of those that the user is allowed to view, depending upon the function.
backUrl	Used with the two password change-related functions, this provides the URL for a link to go back to after the password change is made.	Single-valued, a string. Default: none.
changeRequestAttr	Use this parameter to name the attribute whose value you want to change. This is the schema name of the attribute, not the display name.	Required. Single-valued, a string. Default: none.
changeRequestType	Use this parameter to describe whether the request is to add or remove information. It has two values: newval remove	Required. Single-valued, a string, one of the listed values. Default: none.

Parameters used for Portal Functions

Parameter Name	Description	Rules
comp	Use this parameter to make sure the page returned shows only the component you requested and nothing more. For example, this omits the navigation bar. If comp is set to true, it will be considered true for the rest of the session, even if not explicitly set in the URL. This parameter is optional for all functions that use it, but strongly recommended.	Single-valued, Boolean, true or false. Default: false.
coords	These are present if the user clicked on the location map.	Single-valued, a text string representing a pair of coordinates, presented as xx, yy. Default: none. Either coords or rectangle may be part of the URL, but not both.
days	Use this parameter to specify a limited window, n days back from the current time, within which to look for requests.	Single-valued, an integer >=1. Default: 0, meaning no limit; look as far back as the oldest request.
displayFormat	Use this parameter to specify the type of view for the results.	Single-valued, an integer 2 - use table format. 3 - use custom format.
expandAllGroups	Use this parameter to expand all groups that you have rights to expand. If set to true, then all such groups are expanded. If set to false, then only the groups specified with the groupsToExpand parameter are expanded.	Single-valued, Boolean, true or false. Default: false.
graphviewtype	Use this parameter to specify the format of an organization chart. There are two possibilities: 1—A vertical presentation, with parents above children. 2—A horizontal layout, with parents to the left of children.	Single-valued. Default: 1.

Parameters used for Portal Functions

Parameter Name	Description	Rules
groupsToExpand	Use this parameter to specify one or more target groups you want to expand.	Multi-valued, a DN. Default: none.
locId	Location at which the object resides.	Single-valued, a DN. Default: none.
locObjClass	The location objectclass name.	Single-valued. Default: oblixlocation.
login	The identifying string of characters provided by the user, along with the password, to log in. This is usually some variation on the user name.	Single-valued. Default: none.
noOfFields	Use this parameter to specify the number of attributes whose values are to be searched through. Depending on the value of this parameter, you must provide the same number of sets of STy, SLk and SSt parameters. For example, if the noOfFields is 2, then required parameters would be STy1, SLk1 and SSt1 and STy2, SLk2 and SSt2. The result of the search is an AND that satisfies all of the parameter sets. The value of noOfFields must be greater than or equal to the number of sets. If it is greater, no error is reported, and the behavior will be just as if you had entered the correct, smaller value for n.	Single-valued, an integer value n >= 1. Default: 1.

Parameters used for Portal Functions

Parameter Name	Description	Rules
noOfRecords	Use this parameter to specify a maximum number of entries to be returned in the search results. This parameter and its default values are overridden by the showAllResults parameter. Note the default is derived from the defaultDisplayResultVar parameter in the oblixbaseparams.xml file. (See the Appendix “NetPoint Parameter Files” on page 193.) However, there is one exception. When a predefined report is created, the report definition includes the number of records to be displayed. This takes its value from the default in effect when the report is generated, and cannot be modified.	Single-valued, an integer value n >=1. Defaults to the value of the defaultDisplay ResultVal parameter.
ObWfComment	Use this parameter to provide a comment for a step in a workflow.	Single-valued, string. Default: none.
ObDomainName	Use this parameter to specify the name of the domain in which you want to create, change, or remove an entry. The domain name must be defined under the workflow referred to by the ObWorkflowName parameter.	Single-valued, a DN. Default: none.
ObWorkflowName	Use this parameter to specify the name of the workflow that you want to use to create, change, or delete a directory entry.	Single-valued, a DN Default: none.
portalid	Use this parameter to specify a label that applies to a combination of a backURL, button image, and mousover text that has been added to the PICI file. The label entered persists for the rest of the session, meaning that it continues to apply as though the parameter had been used in successive URLs. Use the value none to end persistence	Any text. Default: none.

Parameters used for Portal Functions

Parameter Name	Description	Rules
rectangle	Rectangle on the map indicating the location of the object.	Single-valued, a text string holding two pairs of numbers, the coordinates of the upper left and lower right corners of the rectangle, in the form xx1,yy1:xx2,yy2. Default: (If not given, only the object's map is shown with the location of the object on that). Either coords or rectangle may be part of the URL, but not both.
reportname	Use this parameter to provide the name of an existing report.	Single-valued. Default: none
reportsubtab	Use this parameter to specify that you want the results of an existing report. Currently, the only legal value is predefinedreport.	Single-valued. Default: none.
requestType	Use this parameter to specify which of the two possible request queue types you want to search. incomingRequests—Requests you need to process. outgoingRequests—Requests you have originated.	Single-valued. Default: none.
scoperesolved	If rectangle is specified, then scopeResolved must be set to true.	Single-valued, Boolean, true or false. Default: if this parameter is not given, the scope is resolved again.

Parameters used for Portal Functions

Parameter Name	Description	Rules
showAdministratorOfGroups	Use this parameter to ask for groups for which you or another user serve as administrator.	Single-valued, Boolean, true or false. Default: false.
show_all	If set to true, displays all users on the location map.	Single-valued, Boolean, true or false. Default: false.
showAllResults	Use this parameter to force all results of the search to be returned to the user. If the parameter value is true, it overrides the value of the noOfRecords parameter.	Single-valued, Boolean, valued true or false. Default: false, meaning return results up to the limit imposed by the noOfRecords parameter.
showDynamicGroups	Use this parameter to ask to be included in the response to groups in which you or another user serve as dynamic members.	Single-valued, Boolean, true or false. Default: false.
showDynamicUserMembers	Use this parameter to specify whether dynamic members of a group are to be included in the response.	Single-valued, Boolean, true or false. Default: false.
showMemberOfGroups	Use this parameter to ask to be included in the output of groups in which you or another user serve as members.	Single-valued, Boolean, true or false. Default: false.
showNestedGroups	Use this parameter to ask for nested groups you, or another user, are a member of to be included in the response.	Single-valued, Boolean, true or false. Default: false.

Parameters used for Portal Functions

Parameter Name	Description	Rules
showNestedUserMembers	Use this parameter to specify whether nested members of a group are to be included in the response.	Single-valued, Boolean, true or false. Default: false.
showOwnerOfGroups	Use this parameter to ask for groups you, or another user, are an owner of to be included in the output.	Single-valued, Boolean, true or false. Default: false.
showStaticGroups	Use this parameter to ask for groups you, or another user, are a static member of to be included in the response.	Single-valued, Boolean, true or false. Default: false.
showStaticUserMembers	Use this parameter to specify whether static members of a group are to be included in the response.	Single-valued, Boolean, true or false. Default: false.

Parameters used for Portal Functions

Parameter Name	Description	Rules
SLkn	Use this parameter to choose the way string data is selected. Legal entries all begin with the letter O, and the next two letters are an abbreviation of the search type. Possible values are: OSM—Substring match. Search results include entries whose value contains the exact data entered for this parameter, including spaces. OGE—Greater than or equal to. Search results include entries whose string value is greater than or equal to the data entered for this parameter. OLE—Less than or equal to. Search results include entries whose string value is less than or equal to the data entered for this parameter. OBW—Begins with. Search results include entries whose string value begins with the data entered for this parameter. OEW—Ends with. Search results include entries whose string value ends with the data entered for this parameter. OSL—Sounds like. Attempts a phonic match on the entered data. OEM—Exact match. Search results include entries whose string value is the same as the data entered for this parameter. OOS—Oblix-specific substring match. Differs from OSM. Spaces are considered to be delimiters, and results include entries which match both of the two strings. Any other value than the ones specified above returns an error (Invalid parameters).	Multi-valued, 1 to n. For an explanation of n, see noOfFields. Default: none. If an invalid value or no value is provided, an error is returned.
sortBy	Use this parameter to specify which one of the attributes to use to sort the results. Use the schema name, not the display name.	Single-valued. Default: if no value is specified, the class attribute of the structural objectclass of the tab specified by tab_id is used.

Parameters used for Portal Functions

Parameter Name	Description	Rules
sortOrder	Use this parameter to specify the sort order, ascending or descending. There are two possible values. ascending descending	Single-valued. Default: ascending.
SStn	Use this parameter to provide a string value to be searched for. Note: The value provided for this parameter must be equal to or greater than the value of SearchStringMinimumLength in the userservcenterparams.xml file. (See “NetPoint Parameter Files” on page 193.)	Multi-valued, 1 to n. For an explanation of n, see the noOfFields parameter. Default: If no value is specified, then the default is to do a blank search on the class attribute. This means, return everything that has any value (other than a NULL value) for the selected STy attribute.
startFrom	Use this parameter, for a long list of search results, to skip over a selected number of items and start the list with a specified item. For example, if 100 entries were found by the search, entering a value of 80 for this parameter gives a response showing only items 80 through 100.	Single-valued, integer. Default: 0, meaning to start from the beginning of the search results list.
STyn	Use this parameter to specify an attribute whose string values are to be searched. Attributes are associated, by application, with one or more tabs. The attribute must have been marked as searchable for the tab name provided or assumed for the tab_id parameter. If it is not, an error is returned. An administrator must have set the searchable flag for the attribute.	Multi-valued, 1 to n. For an explanation of n, see the noOfFields parameter. Default: none.

Parameters used for Portal Functions

Parameter Name	Description	Rules
tab_id	Use this parameter to specify the name of the tab which describes the information category you want to work within. Possible values for the parameter differ across applications. For User Manager and Group Manager, only one tab is allowed. For Organization Manager multiple tabs are allowed. If omitted, NetPoint can always find a default value for tab_id, as described in the Rules column. However, Oblix recommends you always provide a value for tab_id. This will provide self-documentation for each portal link you create and give you exactly what you want regardless of what other changes might be made to the system. For example, Organization Manager allows you to change the order in which tabs are displayed. If you rely on the default tab_id in this case, all your portal functions would be affected and might not work correctly.	Single-valued. Default: For User Manager and Group Manager, which have only a single tab, tab_id is assumed. For Organization Manager, which has multiple tabs, the tab_id is assumed to be that for the leftmost tab.
target	Determines the window in which the page is displayed. It takes two possible values: self—Displays in the same window from which it was called. top—Displays in the top browser window.	Single-valued, a string. Default: self.
targetApplication	Use this parameter to specify the application to be searched for tickets. If you want to search all applications, use the value allApplications. To search a specific application, enter the internal application name: groupservcenter—For Group Manager objservcenter—For Organization Manager userservcenter—For User Manager	Single-valued. Default: none.

Parameters used for Portal Functions

Parameter Name	Description	Rules
ticketType	Use this parameter to specify the status type for the requests to be searched for. There are three possible entries. WfAllTickets—Search for all requests, regardless of status. WfCompletedTickets—Search for requests that have been completely processed. WfPendingTickets—Search for requests that are pending, only partially processed.	Single-valued. Default: none.
uid	Use this parameter to specify the DN of an entry you want to view or modify. NOTE: This parameter is used in many functions, and is NOT limited to User Manager activities, which might be assumed from its name.	Single-valued, a DN. Default: none.
workflowInstanceDn	Use this parameter to specify the DN of the workflow for which information is required. To specify the step for which the information is required, provide the workflowStepInstanceId parameter. The DN for the workflow is shown in the workflow definition view (see the *NetPoint 7.0 Administration Guide Volume 2*).	Single-valued, a DN. Default: none.
workflowStepInstanceId	Use this parameter to specify a certain step, in the workflow specified by workflowInstanceDn, for which information is required.	Single-valued, integer value Default: none

Portal Inserts Example

This section illustrates one method of providing a NetPoint portal to users.

Task overview: One method of providing a NetPoint portal

1. Identify the NetPoint function(s) you intend to provide.
2. For each function, use the descriptions from this chapter to determine the URL components required to make the request to NetPoint, then construct the URL.

3. Develop a Web page in HTML to serve as a starting point for the NetPoint functionality you are providing. This page will contain links and/or forms that access NetPoint as a portal.
4. Deploy your page on the intranet.
5. Distribute the URL of your index page to users.

This example builds a portal insert that displays the COREid System profile page from NetPoint User Manager for a given user using User Manager's `view` function.

Collect the following information for the URL:

Parameter	Description
`uid`	DN of the user whose profile you wish to view, this month's Star Employee, taken from the directory. (The index page administrator updates this each month).
`attrName`	Attributes to be returned. This parameter is used repeatedly to ask NetPoint to return the Full Name, Photo, Email, Title and Phone Number attributes, so that viewers can write or call to congratulate Star Employees on their achievement.

For the example the base URL for User Manager is:

```
http://techpubs.com:88/identity/oblix/apps/
          userservcenter/bin/userservcenter.cgi
```

You manually learn from Human Resources, or an external system, who is this month's Star Employee, and locate them in the directory to get their DN. The DN in the example is:

```
cn=Rohit Valiveti,ou=Sales,ou=Dealer1k1,
          ou=Latin America,ou=Ford,o=Company,c=US
```

Applying URL encoding to this to escape special characters like = and space gives:

```
cn%3DRohit%20Valiveti%2Cou%3DSales%2Cou%3DDealer1k1%2C
ou%3DLatin%20America%2Cou%3DFord%2Co%3DCompany%2Cc%3DUS
```

which is used for the value of the `uid` parameter in the URL.

In the example directory, the attribute names needed for the profile data you have decided to show in the profile page are as follows:

- cn (Full Name)
- genbadgephoto (Photo)
- description
- mail (Email address)

- genphonenumber (phone number)
- title (individual's title)

Note: These attribute names are very likely to be different in your actual deployment environment; you must check the directory schema or ask the directory administrator for the names in use in your target environment.

Finally, set `comp` to true, to exclude the navigation bar and search form from the result and display only the component.

You now have all the information required to construct the following URL:

```
http://techpubs.com:88/identity/oblix/apps/userservcenter/
bin/userservcenter.cgi
        ?program=view
             &uid=cn%3DRohit%20Valiveti%2Cou%3DSales%2Cou%
                3DDealer1k1%2Cou%3DLatin%20America%2Cou%3
                DFord%2Co%3DCompany%2Cc%3DUS
             &attrName=cn&attrName=genbadgephoto
             &attrname=description
             &attrName=mail&attrName=genphonenumber
             &attrname=title
             &comp=true
```

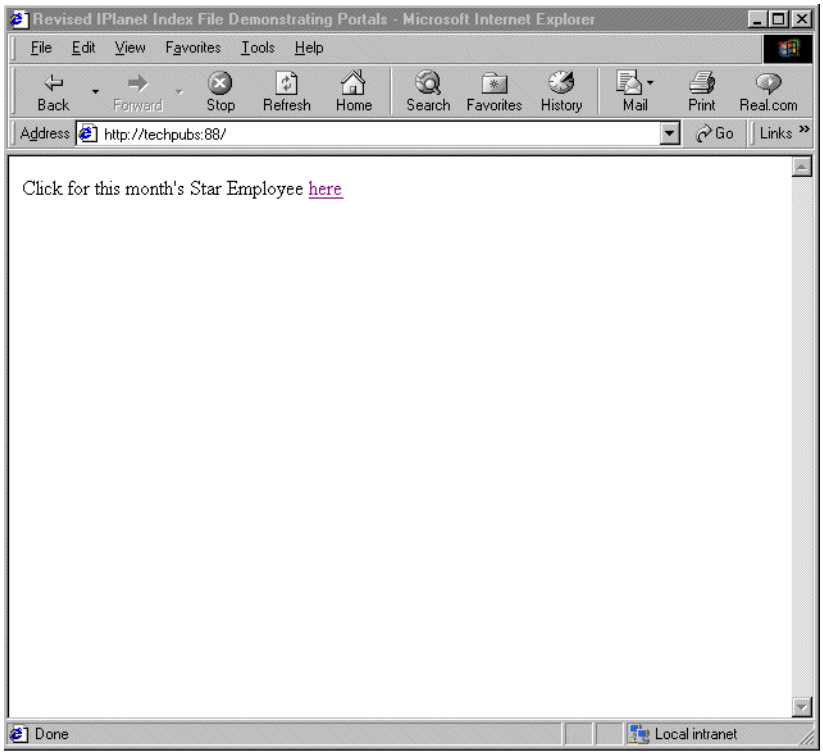
Now you can develop a simple Web page with a link to the view function. You are of course free to design any page you like. As long as it can generate a similar URL request, NetPoint does not care how you did it.

For this example, you can replace the index page for the Web server through which you access NetPoint with the following HTML:

```
<html>
<head>
   <title>
        Revised iplanet Index File Demonstrating Portals
   </title>
</head>
<body>
   <p>
        Click for this month's Star Employee
        <a href="portalsexamples.html">
        here
        </a>
     </p>
```

```
</body>
</html>
```

Now, users trying to access the Web server at the URL http://techpubs:88 see the following page instead of the iPlanet index page:



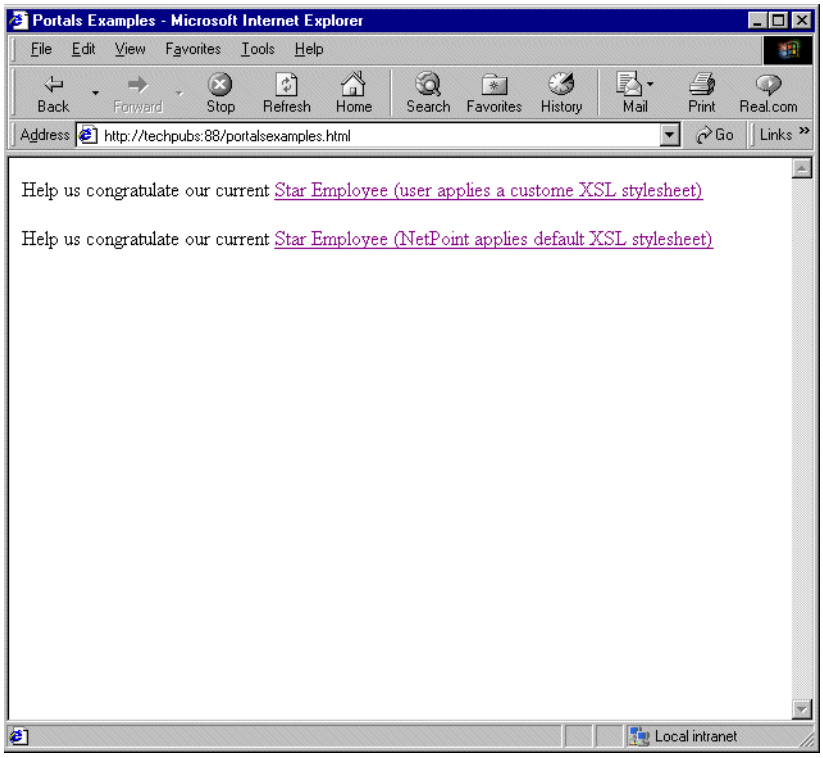
The users click on the link to the `portalsexamples.html` page, whose content is the following:

```
<html>
<head>
   <title>Portals Examples </title>
</head>
<body>
   <p>Help us congratulate our current
         <a href="http://techpubs:88/identity/
            oblix/apps/userservcenter/bin/
            userservcenter.cgi
               ?program=view
         &uid=cn%3DRohit%20Valiveti%2Cou%3DSales
            %2Cou%3DDealer1k1%2Cou%3DLatin%20America
            %2Cou%3DFord%2Co%3DCompany%2Cc%3DUS
         &attrName=cn
```

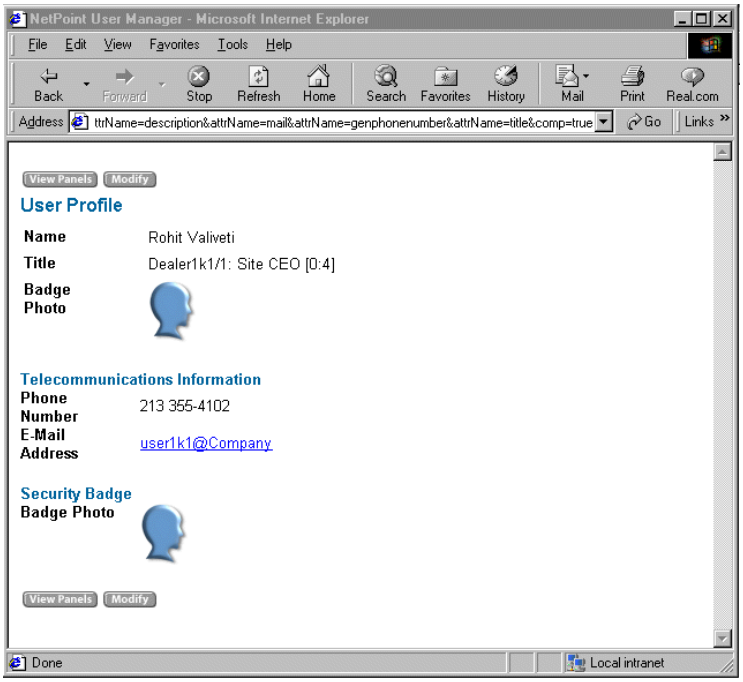
```
            &attrName=genBadgePhoto
            &attrName=description
            &attrName=mail&attrName=genphonenumber
            &attrName=title
            &comp=true
            &xsl=usc_profilenew.xsl
            ">
            Star Employee (user applies a custom XSL stylesheet)
            </a>
      </p>

         <p>Help us congratulate our current
            <a href="http://techpubs:88/identity/
               oblix/apps/userservcenter/bin/
               userservcenter.cgi
            ?program=view
            &uid=cn%3DRohit%20Valiveti%2Cou%3DSales
            %2Cou%3DDealer1k1%2Cou%3DLatin%20America
            %2Cou%3DFord%2Co%3DCompany%2Cc%3DUS
            &attrName=cn
            &attrName=genBadgePhoto
            &attrName=description
            &attrName=mail&attrName=genphonenumber
            &attrName=title
            &comp=true
            ">
            Star Employee (NetPoint applies default XSL
               stylesheet)
            </a>
      </p>
   </body>
   </html>
```

They see this page:



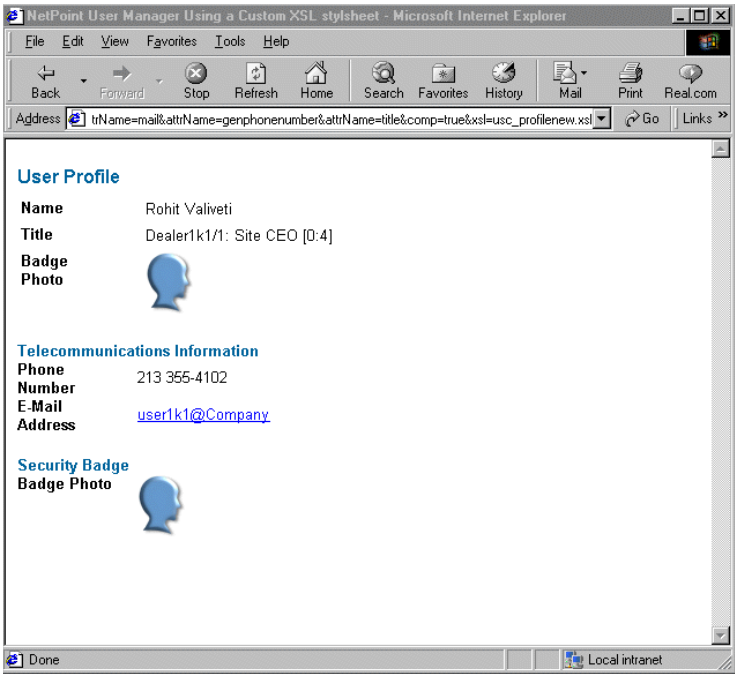
Then, depending upon which link they select, they get two different presentations of the view page. If they click the second link, in which case the default XSL stylesheet is applied, they will see the following page. Note that the View Panels and Modify buttons appear on this page.



If the user clicks the second link, NetPoint uses a modified version of the stylesheet that controls the displayed content for this page. See the extra parameter:

&xsl=usc_profilenew.xsl

provided in the URL. This stylesheet expressly removes the buttons from the presentation. It also modifies the title displayed in the browser window, as seen in the following example. Methods for creating this custom stylesheet are discussed in “Designing the GUI with PresentationXML” on page 15.


NetPoint User Manager Using a Custom XSL stylsheet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Real.com
Address trName=mail&attrName=genphonenumber&attrName=title&comp=true&xsl=usc_profilenew.xsl
Go Links
User Profile
Name
Rohit Valiveti
Title
Dealer1k1/1: Site CEO [0:4]
Badge Photo
Telecommunications Information
Phone Number
213 355-4102
E-Mail Address
user1k1@Company
Security Badge
Badge Photo
Done
Local intranet

4 Modifying Catalog Files

NetPoint makes extensive use of *catalog files* to configure various system attributes and behaviors. Catalog files are files ending with param.xml or msg.xml. These files control NetPoint behavior and message content, respectively. This chapter describes some of the many changes that can be made to these files. Topics include:

A description of the param.xml file structure and contents, and a discussion of how to change them, is provided in "NetPoint Parameter Files" on page 193.

Setting Overall and Attribute Specific Date Formats

Problem: When you install NetPoint for the first time, a default format is used for all attributes that are configured to use a display type "Date". This default display format cannot be changed during installation. Later, you may need to change the default format, or set it differently for different attributes.

Solution: Change date formats in either of two ways, described in this section:

- Change the universal default value. This format is automatically used when attributes are configured to use display type “date”.

  or,

- Change the value stored in the directory for each attribute. The stored format is used instead of the default format.

NetPoint supports the date display type formats appearing in the table below, to control handling of month (M), day (D), and year (Y). Use the format name shown in the data type column to make dates display in the format shown in the Example column. You can also specify a date separator to be used between the MDY values. The date separator can be any of the values “/”. “-”, or ““ (no separator).

dateType Value	Example	Description
`ObIntegerDate`	946080000	Number of seconds after midnight of December 31, 1970
`ObMDYDate`	12/31/1999	mm/dd/yyyy
`ObDMYDate`	31/12/1999	dd/mm/yyyy
`ObDMonthYDate`	31-Dec-1999	dd-MMM-yyyy
`ObMonthDYDate`	Dec-31-1999	MMM-dd-yyyy
`ObISO8061Date`	19993112	yyyyddmm

Modifying Default Date Display

You need to modify the obDateSep and obDateType parameters in the globalparams.xml file, as described below.

To modify the default date display type for the system

1. Open the globalparams.xml file using a text editor.

   `CORElid_install_dir``/identity/oblix/apps/common/bin/globalparams.xml file`

2. Locate the lines to be changed (dash (-) and slash (/) are supported date separators).

   For example:

```
<SimpleList>
  <NameValPair
    ParamName="obDateSep"
    Value="/"></NameValPair>
</SimpleList>
<SimpleList>
  <NameValPair
    ParamName="obDateType"
```

```
      Value="ObMDYDate" /></NameValPair>
</SimpleList>
```

3. Change the value for the `obDateType` parameter, as needed, using one of the `obDateType` values from the table, then change the value for the `obDateSep` parameter.

   For example, the changed lines might look like this:

   ```
   <NameValPair ParamName="obDateSep" Value="-" />
   <NameValPair ParamName="obDateType" Value="ObMonthDYDate"/>
   ```

4. Save your changes and close the file.
5. Stop and start the COREid Server.

Modifying Date Display by Attribute

Date formats can also be set for specific attributes. Within the COREid System, go to COREid System Configuration > User Manager > Common Configuration > Configure Object Class, and select the class whose attribute is to be modified. Under the View Object Class screen, select Modify Attributes. For any attribute of display type `date`, you are shown two fields, the date type and the date separator. Change the content of these fields to match the values you want to use.

Setting the Date Range for the Year Drop-Down List

Problem: By default, NetPoint provides a drop-down list of years ranging from 1993 to 2012. This is reasonable when the dates entered are those for badge issue or expiration dates, but is not useful if the date calls for a bigger range, such as one to include a birth date.

Solution: Extend the year date range that is carried in the basic.xsl stylesheet. For a more detailed explanation of why this works, see “Designing the GUI with PresentationXML” on page 15.

Note: The standard NetPoint XSL files define only this one dropdown year date range. The method described here will have a global effect. The changes described below *must* be made to both of these files.

To set the date range for the year drop-down list

1. Locate the basic.xsl stylesheet in the directory below:

   COREid_install_dir/identity/oblix/lang/shared/basic.xsl

   where *COREid_install_dir* is the directory where COREid is installed, *langTag* is a language tag (en-us for example) in RFC 1766 format.

2. Copy the stylesheet to your custom directory to overwrite the wrapper file there.

   For example:

   COREid_install_dir/identity/oblix/lang/*langTag*/*Custom_Dir*/basic.xsl

3. In the stylesheet in your custom directory, locate the templates for ObDateYear.

4. Locate the lines in the templates that begin and end the definition of the year drop-down list.

For example, the year drop-down list is defined by the lines below: `<xsl:template name="ObDateYear">`

```
...
- <xsl:if test="@obyear">
- <option value="{@obyear}" selected="true">
  <xsl:value-of select="@obyear" />
  </option>
  </xsl:if>
- <xsl:if test="@obyear">
  - <option value="{@obyear}" selected="true">
        <xsl:value-of select="@obyear" />
    </option>
  </xsl:if>
  <option value="">----</option>
  <option value="1993">1993</option>
  <option value="1994">1994</option>
  <option value="1995">1995</option>
  ...
  <option value="2012">2012</option>
  </select>
</xsl:template>
```

5. Copy, paste, and edit the pattern to extend the year date range.

   For example, to include dates before 1993 the pattern goes between the lines shown below:

```
<option value="">----</option>
<option value="1992">1992</option>
<option value="1993">1993</option>
```

to extend dates beyond 2012, insert lines as shown below:

```
<option value="2012">2012</option>
<option value="2013">2013</option>
</select
</xsl:template>
```

6. Copy the stylesheet to your custom style directory on the WebPass.

   WebPass_install_dir/identity/oblix/lang/*langTag*/*Custom_Dir*/basic.xsl

7. Stop and start the COREid Server and WebPass

Changing the Color of the Configure Attributes Panel

Problem: Users may need to change foreground and background colors for the Configure Attributes panel.

Solution: Change RGB values that control these colors in the `oblixadminparams.xml` file.

To change the Configure Attributes panel color

1. Locate the `oblixadminparams.xml` file in the directory:

COREid_install_dir/identity/oblix/apps/common/bin/oblixadminparams.xml

   where *COREid_install_dir* is the directory where COREid is installed.

2. Locate the entries:

```
<NameValPair ParamName="config_meta_attr_applet_bg"Value="cccccc" />
<NameValPair ParamName="config_meta_attr_applet_fg"Value="000000" />
```

   The parameter bg controls background color, fg controls foreground. Values following the colon are RGB hex values for color.

3. Change the RGB values to set the new colors.
4. Save and close the file.
5. Restart your COREid Server for the changes to take effect.

Changing Top Navigation Bar Application Name

Problem: You want to change the mouseover text used for COREid System top Navigation Bar buttons.

Solution: Change the mouseover button text, controlled by the oblixbaseparams.xml file.

To change the top navigation bar application name

1. Locate the oblixbaseparams.xml file in the directory:

   COREid_install_dir/identity/oblix/apps/common/bin/oblixbaseparams.xml

   where *COREid_install_dir* is the directory where COREid is installed.

2. In this file, locate the controlling text for any of the modules.

   For example, the text for Group Manager:

```
</ValNameList>
<ValNameList ListName="groupservcenter_application_info">
  <NameValPair ParamName="VERSION" Value="7.00" />
  <NameValPair ParamName="CODE" Value="GM70" />
  <NameValPair ParamName="ID" Value="groupservcenter" />
 <NameValPair ParamName="PROGRAM" Value="../../groupservcenter/
bin/groupservcenter.cgi" />
  <NameValPair ParamName="DESCRIPTION" Value="Group Manager" />
 <NameValPair ParamName="NAVBAR_GIF" Value="T1TABgroupmanager"
/>
 <NameValPair ParamName="NAVBAR_GIF2" Value="T1TABgroupmanager"
/>
 <NameValPair ParamName="NAVBAR_GIFDIR" Value="../../common/ui/
style0/" />
  <NameValPair ParamName="WORKFLOW_ALLOWED" Value="true" />
  </ValNameList>
```

   The text following `DESCRIPTION` is the information that displays when you place the mouse pointer over the NetPoint Group Manager button.

3. Change the text entry to the content needed.
4. Save and close the file.
5. Restart your COREid Server for the change to take effect.

Changing User Name/Password Text on Logon Screen

Problem: The default NetPoint login screen shows two text fields, `Username` and `Password`. Some administrators want to change those text labels to a different value; for example, to show `UID` rather than `Username` as the login name.

Solution: Change the text controlled by the `oblixbasemsg.xml` file.

To change user name and password text on the Logon screen

1. Locate and open the oblixbasemsg.xml file in the directory:

   `COREid_install_dir``/identity/oblix/lang/`*`langTag`*`/oblixbasemsg.xml`

   where *COREid_install_dir* is the directory where COREid is installed and *langTag* is a language tag (en-us, for example) in RFC 1766 format.

   The oblixbasemsg.xml file contains paired sets of data, in the form:

   <Message MsgTag="*message name*">*Message text*</Message>

   NetPoint uses "*message name*" to locate the text that is to be displayed.

   Note: Do not change the "*message name*".

   NetPoint displays *Message text*, which is variable text and can be changed.

2. Locate the *Message text* that you want to change.

   For example:

   ```
   <Message MsgTag="MUsername">Username</Message>
   <Message MsgTag="MPassword">Password</Message>
   ```

3. Change the message text:

   For example, to change MUsername to UID:

   ```
   <Message MsgTag="MUsername">UID</Message>
   ```

4. Save and close the file.

5. Restart your COREid Server for the change to take effect.

Changing Parameter Catalogs to Control Operation

Problem: You want to change NetPoint operation in ways that are not specifically called out in this Guide.

Solution: NetPoint is controlled primarily by hardcoded logic. At some points, generally at the user interface, the content of certain text files guides operation. You can change the content of some of these files.

A description for how to do this, and a list of the parameter catalogs and their changeable content, is provided in “NetPoint Parameter Files” on page 193.

Changing Message Catalogs and MouseOver Text

Problem: You want to revise mouseover text, or change the content of a displayed error message.

Solution: Change text controlled in the message catalog.

As discussed elsewhere, multiple languages are available for use with NetPoint 7.0. Messages that were once in stylesheets are language dependent and are now defined separately as variables in message catalogs. The new NetPoint directory structure consolidates all message catalogs for JavaScript files, XSL, and HTML.

- Any language-specific files will be located in \lang*langTag*.
- Any non-language specific objects are located within \lang\shared.

All the stylesheets have a language-specific wrapper in \lang*langTag*\style0 which *includes* the main language-neutral version stylesheet in \lang\shared. This new wrapper segregates the main stylesheet functionality, which is language independent, from language-specific messages.

Language-specific messages are referred to through variables in message catalog files, as discussed below:

- “Handling Language-Specific Stylesheet Messages” on page 149
- “Handling Language-Specific Messages for JavaScript” on page 150

Handling Language-Specific Stylesheet Messages

The messages for stylesheets are defined in the message catalog below:

COREid_install_dir\identity\oblix\lang*langTag*\msgctlg.xsl

You need to ensure that all displayable strings in your older version stylesheets are placed in the NetPoint 7.0 stylesheet message catalog. For example, suppose you have customized a NetPoint 6.1 stylesheet, navbar.xsl, in:

COREid_install_dir\identity\oblix\apps\common\ui\style0\navbar.xsl

where a message reads as:

```
<xsl:text> &lt;&lt; Click here to return to the previous
application(s). </xsl:text>
```

In the NetPoint 7.0 version of the stylesheet:

COREid_install_dir\identity\oblix\lang\shared\navbar.xsl

you need to modify the message to read:

```
<xsl:text> &lt;&lt; <xsl:value-of select="$MPrevAppln"/> </
xsl:text>
```

and ensure that MPrevAppln is defined in the NetPoint 7.0 message catalog:

*COREid_install_dir**install_dir*\identity\oblix\lang*langTag*\msgctlg.xsl

as follows:

```
<xsl:variable name="MPrevAppln">Click here to return to the
previous application(s). </xsl:variable>
```

To handle language-specific message catalogs for XSL stylesheets

1. Locate the stylesheet containing the message.

   For example:

   COREid_install_dir\identity\oblix\lang\shared\navbar.xsl

   ```
   <xsl:text> &lt;&lt; Click here to return to the previous
   application(s). </xsl:text>
   ```

2. Copy the stylesheet to the custom style directory.

   For example:

   COREid_install_dir\identity\oblix\lang*langTag**Custom_dir*\navbar.xsl

3. Modify the message in the stylesheet to use the appropriate message catalog parameter.

   For example:

```
<xsl:text> &lt;&lt; <xsl:value-of select="$MPrevAppln"/> </
xsl:text>
```

4. Locate the language-specific message catalog.

   COREid_install_dir\identity\oblix\lang*langTag*\msgctlg.xsl

5. Ensure that the message parameter is properly defined.

   ```
   <xsl:variable name="MPrevAppln">Click here to return to the
   previous application(s). </xsl:variable>
   ```

6. Restart your COREid Server to have any changes to take effect.

Handling Language-Specific Messages for JavaScript

Pop-up messages in JavaScript files are also replaced by variables. The message catalog for JavaScript files is located in:

WebPass_install_dir\identity\oblix\lang*langTag*\msgctlg.js

Each language-specific message catalog is divided into sections that show the messages for specific JavaScript files, several of which are named below:

misc.js
miscs.js
monitorwf.js
personselector.js
proxyadmin.js
selector.js
atickets.js
wfqs.js
deactivateuser.js
confirm.js

You need to ensure that all displayable strings are placed in the message catalog and the message catalog must be referenced through the I18N_GetMsg function.

For example, the code in the JavaScript file:

WebPass_install_dir\identity\oblix\lang\shared\admin.js

that pops up a message:

```
alert("Room must have a name.")
```

appears as:

```
alert(I18N_GetMsg('MRoomNameReq'))
```

where MRoomName is defined in:

WebPass_install_dir\identity\oblix\lang*langTag*\msgctlg.js

as:

```
MESSAGE_CATALOG[ 'MRoomNameReq' ] = "Room must have a name.";
```

Note: Oblix recommends that you retain the files in \shared as a reliable backup and instead copy the file to be customized into your custom style directory first.

To handle language-specific message catalogs for JavaScript files

1. Ensure that all displayable strings for JavaScript files are placed in the message catalog:
2. *WebPass_install_dir*\identity\oblix\lang*langTag*\msgctlg.jsEnsure that the message catalog is referenced through the I18N_GetMsg function located in (automatically loaded):

   WebPass_install_dir\identity\oblix\lang\shared\i18n.js
3. Restart your COREid Server and WebPass to have any changes to take effect.

Before NetPoint 6.5

Prior to NetPoint 6.5 and 7.0, messages were controlled by an XML file pertinent to the application. Files were stored in:

COREid_install_dir/identity/oblix//*appname*msg.xml

where *appname* matches a specific application, as follows:

groupservcenter—Group Manager
objservcenter—Organization Manager
userservcenter—User Manager

where *COREid_install_dir* is the directory where COREid is installed and *languageTag* is a language tag in RFC 1766 format.

Each file contains multiple paired sets of data, in the form

```
<Message MsgTag="the tag name">The tag text</Message>
```

Note: Do not change the value for the MsgTag; this is used by the system to locate the text. You may, however, change the tag text.

For example, you can change:

```
<Message MsgTag="OMyRequests">Check my request tickets.
</Message>
```

to:

```
<Message MsgTag="OMyRequests">Check my workflow history.
</Message>
```

Supporting UTF-8 Data

Problem: You want to make NetPoint work with international characters from the UTF-8 character set.

Solution: Follow the steps below to modify the globalparams.xml and globalparams.lst files.

Note: This solution assumes that *all* of your directory data is UTF-8 format. NetPoint *does not* support a mix of data types in the directory.

To import the UTF-8 character set

1. Change the value of the doUtfConversion parameter to Yes (NameValPair ParamName="doUtfConversion" Value="Yes") in the following files:

   For each COREid Server:

   COREid_install_dir/identity/oblix/apps/common/bin/globalparams.xml

   where *COREid_install_dir* is the directory where COREid is installed.

   For each WebPass:

   WebPass_install_dir/identity/oblix/apps/common/bin/globalparams.xml

   where *WebPass_install_dir* is the directory where WebPass is installed.

   For each Access Server:

   AccessServer_install_dir/access/oblix/apps/common/bin/globalparams.lst

   where *AccessServer_install_dir* is the directory where Access Server is installed.

   For each WebGate:

   WebGate_install_dir/access/oblix/apps/common/bin/globalparams.lst

   where *WebGate_install_dir* is the directory where WebGate is installed.

2. Import the UTF8 data.
3. Restart your Web server(s), COREid Server(s) and Access Server(s).
4. Web browsers and other connected NetPoint clients should also be restarted after a character set change, as any cached data would belong to the old character set.

   Note: The character encoding of Web browsers should always be set to Latin1 (ISO-8859-1) regardless of the data format.

   ---

5 Other Customization

This chapter covers various ways you can customize the NetPoint application that do not fall cleanly into any of the categories covered earlier.

This chapter discusses the following:

Customizing to Allow Auto-Login

CoreID supports a self registration workflow. For the User Manager version *only*, this can be used to provide an auto-login capability. This capability can be extended to allow users of the workflow to access additional resources immediately following the self registration, without being challenged. In addition, for COREid resources, these users can be automatically authenticated after Self Registration. This section describes the steps necessary to accomplish this.

Task overview: Customizing NetPoint for auto-login

1. Add at least one AccessGate, as described in the *NetPoint 7.0 Installation Guide*.

   The AccessGate that you add must have its Access Management Service option set to `true`. You add this AccessGate simply to ensure that at least one has been created and configured, to control access to COREid.

2. Configure the AccessGate, as described in the *NetPoint 7.0 Administration Guide Volume 2*.

   From the *COREid_install_dir*/identity/AccessServerSDK/oblix/tools directory, run the configureAccessGate program to configure the AccessGate. No special configuration data needs to be provided to satisfy Auto-login requirements.

   COREid_install_dir is the directory where COREid is installed.

3. Enable auto-login using a certain COREid Parameter file.

   In the *COREid_install_dir*/identity/oblix/data/common/basedbparams.xml file, particular parameter values need to be set, as defined in the following table. You will need to know the URL and access method for the page that the user will go to immediately following self-registration.

Parameter Name	Value
SelfRegGeneratesSSOCookie	true
SR_SSOCookieMethod	Access method for the next page.
doAccessServerFlush	false
enableAllowAccessCache	true
SR_SSOCookieURL	Location of the next page.
SR_SSOCookieDomain	A valid domain name, for example, example.com
SR_SSOCookiePath	Location of the next page.

Here is an example of this file content after these changes have been made.

```
<?xml version="1.0"?>
     <ParamsCtlg xmlns="http://www.oblix.com"
          CtlgName="basedbparams">
     <CompoundList ListName="">
     <SimpleList>
        <NameValPair ParamName="default_policy"
             Value="false"/>
        <NameValPair ParamName="doAccessServerFlush"
           Value="false"/>
        <NameValPair
           ParamName="SelfRegGeneratesSSOCookie"
           Value="true"/>
        <NameValPair ParamName="SR_SSOCookieMethod"
           Value="GET"/>
        <NameValPair ParamName="enableAllowAccessCache"
```

```
      Value="true"/>
    <NameValPair ParamName="SR_SSOCookieURL"
      Value="/identity/oblix"/>
    <NameValPair ParamName="SR_SSOCookieIP"
      Value="192.168.1.109"/>
    <NameValPair ParamName="SR_SSOCookiePath"
      Value="/"/>
    <NameValPair ParamName="SR_SSOCookieDomain"
      Value="example.com"/>
  </SimpleList>
  </CompoundList>
  </ParamsCtlg>
```

4. Protect the URL specified in the SR_CookieURL parameter in the COREid Server basedbparams.xml file with a Basic over LDAP authentication scheme in the Access System.

   If any other type of authentication scheme is used, the ObSSOCookie will not be created for the user and auto-login will fail.

5. Stop and restart the COREid server.

   This step, now or later, is essential in order to load the new content of the parameter file.

6. Configure the WebGate to accept auto-login by using an Access System Parameter Catalog.

   In the file at *WebGate_install_dir*/access/oblix/apps/webgate/WebGateStatic.lst, ensure that either:

   - IPValidation is set to false OR
   - IPValidation is set to true and the value that was used for the IP address (SR_SSOCookieIP) in the basedbparams.xml file is also in the IPValidationExceptions list.

     Since the IP validation is a universally applied parameter and you might want to validate the IP in other cases, the second option is likely the one you will follow.

     If you need to modify the content of the IPValidationExceptions.lst file, locate it at *WebGate_install_dir*/access/oblix/apps/webgate. The installed version of this file looks like this:

```
BEGIN:vCompoundList
          DenyOnNotProtected:false
          CachePragmaHeader:no-cache
          CacheControlHeader:no-cache
          IPValidation:false

     # Set UseIISBuiltinAuthentication to true
     # if you are using MSPassport or Integrated
```

```
# Windows Authentication on this machine
# Otherwise leave it set to false
# Only used for IIS

     UseIISBuiltinAuthentication:false

#IPValidationExceptions:
#  BEGIN:vList
#  10.10.50.101
#  10.10.50.102
#  END:vList

WaitForFailover:-1
END:vCompoundList
```

Leave the value of IPValidation set to false, or set it to true. If you change IPValidation, then also remove the # comment markers for the lines BEGIN through END, and change the example IP addresses to the ones you want to allow.

Note: In NetPoint 6.x, the WaitForFailover parameter has been replaced with the *Access Server Timeout Threshold* parameter that can be accessed through the NetPoint GUI (Access System Console > Access System Configuration > AccessGate Configuration). The WaitForFailover parameter is used only for backward compatibility with NetPoint 5.x. Both the WaitForFailover parameter, and its replacement the Access Server Timeout Threshold parameter, control the TCP/IP timeout between the WebGate and the Access Servers it communicates with. The default value is '-1', which means the network default TCP/IP timeout value is used. Be sure that both the WaitForFailover parameter in the webgatestatic.lst file, and its replacement the Access Server Timeout Threshold parameter, use the same value.

---

7. Following the change(s), stop and restart the Web server associated with the WebGate.

   This step, now or later, is essential in order to load the new content of the WebGate parameter file.

To ensure that the URL specified in basedbparams.xml is protected by a policy domain in the Access System, copy the URL and paste it in the Access Tester. If the URL is not recognized, you may need to preface it with the correct host identifier. For instance, the URL of /identity/oblix may not be protected, but the URL //12345:99/identity/oblix might be.

Finally, you must configure a Self Registration workflow in User Manager, as follows.

Task overview: Configuring a Self Registration workflow

1. Create a workflow, as described in the *NetPoint 7.0 Administration Guide Volume 1*.

   The first step of the workflow must be Self Registration. This step must have the login and password semantic type attributes configured. The password attribute must be part of the self-registration step for auto-login to work.

2. Optionally, you can add non-interactive steps, such as external actions. Interactive steps in the workflow will be impossible, because the user is not considered logged in until after the workflow completes.

3. Configure Enable as the last step.

This workflow will generate SSO cookies that can be used for authentication if the workflow completes successfully. In addition, the new user can use COREid without having to log in.

If you need it, the cookie generated as part of auto-login can be obtained from the auto-login page output using IdentityXML. Look for the cookie under the ObSSOCookie element and extract the data for ObValue.

The pertinent part of the XML string will look something like:

```
<ObSSOCookie>
<ObDisplay obdisplayName="SSOCookie"
        obdisplayType="textS" obname="ObSSOCookie"
        obmode="view" obcanRequest="false"
        obrequired="false">
   <ObTextS>
   <ObValue>
        ghv6XNmGZefMq8cgIte08alq477M
        nvaivG+tSaxHRzaOXBcfMkzmf/
        UeTrKcg2jJmyo3PpNbLKS+UgmRi/
        rg8Ac2LlU9a7rprYjqocs
        QGQEgymqELZC0VQo6KqGguv7ujrBt9JtzwQ6/
        sDJFlVaLDwLs0vJbg5kop5

        FASBF9ohGOwQcUtDGlVal
        DwktElNskHYgtjvjc9pBBGtlU9sGuYA/cTw==
   </ObValue>
   </ObTextS>
</ObDisplay>
</ObSSOCookie>
```

Setting Up Self Registration Through IDXML

If the Web pass is protected by a WebGate, you can set up self registration through IDXML for those users who can register themselves. To avoid lockout conditions, you may want to allow self-registration for NetPoint administrators and for users who lost their passwords. You use the following URL for the self-registration request:

```
/identity/oblix/apps/userservcenter/bin/
userservcenter.cgi?/from_prog=workflowSelfRegistration
```

The URL for self-registration is not the usual /identity/oblix/apps/userservcenter/bin/userservcenter.cgi.

Customizing Logout

As described in the *NetPoint 7.0 Administration Guide Volume 2*, NetPoint provides a way to specify an single sign-on logout URL. As part of the installation process, NetPoint automatically sets up the logout URL

/access/oblix/apps/common/bin/logout.html

Starting with NetPoint 6.5, for WebGate, the logout URL is configurable in the LogOutUrls property of WebGateStatic.lst.

The logout.html file activates JavaScripts which perform the actual logout. Users may change this to a different URL, which would for example activate an HTML, CGI or even a PERL file created by the user.

Task overview: Customizing logout

1. Create a different HTML or CGI file to perform the logout steps. This file must have the characters logout. somewhere in its name, for example, mybanklogout.cgi.
2. Store it in a defined location. Oblix recommends using /access/oblix/apps/common/bin.
3. Within the Access System, navigate to Access System Console > System Configuration > View Server Settings > Configure SSP Logout URL.
4. Replace the default URL with one pointing to the location of your new logout process.
5. Within the Access System, navigate to Access Manager > Create Policy Domain. Create a new policy domain for this logout resource using the NONE authentication scheme. The URL Prefix entered to the Resource page should be one that includes the logout resource file or its the parent folder.

Customizing Workflow Email Notifications

COREid's workflow feature allows the user to associate a pre- or post-notification email with a workflow step, using standard PresentationXML techniques to build the email messages. An OutPutXML data stream is combined with a stylesheet to create a logical file. This file is passed to a Mail Server queue from which it is eventually sent to its final destination. COREid expects to find the necessary stylesheets in the directory:

COREid_install_dir/identity/oblix/lang/*langTag*/style0

where *COREid_install_dir* is the directory where COREid is installed and *langTag* is a language tag in RFC 1766 format.

The Mail Server will have been previously configured to use either the Text-only mail style or the HTML mail style. (The MHTML mail style cannot be customized.) That setting controls the choice of the default stylesheet. The default stylesheet is wf_prepostnotification_txt, if the mail style is set to Text-only or wf_prepostnotification_html if the mail style is set to HTML. If an error occurs during workflow processing, CORE Id uses the stylesheets wf_errornotification_txt for mail style Text-only and wf_errornotification_html for mail style HTML.

Users can expand this functionality to provide distinct message formats for pre- and post-notification, and distinct message formats by workflow ID and the step number within the workflow. To do this, users add new stylesheets to the directory. COREid looks for these files first and if they are present uses them instead of the default files.

The file names added by the user must follow this naming structure:

WfDefId_WfStepDefId_preorpostnotify_mailtype

Following are the meanings of each part of the name:

WfDefId—This is the Id for the workflow for which the notification is to be sent. It is the rdn of the workflow. The ID is automatically created by NetPoint when a workflow is created. You can obtain the ID by using the 'View' feature in the work flow definition screen.

WfStepDefId—This is the workflow step for which the notification is to be sent. It is a number.

preorpostnotify—This is the exact text prenotify or postnotify.

mailtype—This is the exact text txt or html. Understand that this suffix does not determine whether the message will be sent as text or HTML. It serves solely as a way for COREid to find the correct file. For example, if the Mail Server is configured for Text-only, COREid looks for a file with the suffix txt. In this case, if you have created a stylesheet with the html suffix, COREid does not look for it and uses wf_prepostnotification_txt instead.

Following are some example file names:

`1864aaa89df04422bfd33afcdfb45641`_2_prenotify_txt.xsl

This provides a customized prenotification email message for step 2 of Workflow `1864aaa89df04422bfd33afcdfb45641`, *if* the Mail Type has been set to Text-only.

`1864aaa89df04422bfd33afcdfb45641`_4_postnotify_html.xsl

This provides a customized postnotification email message for step 4 of Workflow `1864aaa89df04422bfd33afcdfb45641`, *if* the Mail Type has been set to HTML.

Note: You must put the customized email stylesheets in the `style0` directory, even if you have made some other directory the master style directory. Email processing looks for the email stylesheets only in the `style0` directory.

XML Interface and Special Characters

Certain characters in XML files in certain instances require special handling in order to be correctly interpreted by NetPoint.

OutPutXML Files

The current version of the XML standard (see reference page 173) calls for certain characters to be escaped using the & sign followed by additional text. OutPutXML follows this requirement. Users will need to translate these characters when reading from OutPutXML or provide their escaped equivalents when creating OutPutXML.

A table of these characters, with their escaped values, follows.

Table 1 Special Characters in XML

Character	Escaped Representation
& (ampersand)	&amp;
> (greater than)	>

Table 1 Special Characters in XML

Character	Escaped Representation
< (less than)	<
‘ (single apostrophe)	'
“ (double quote)	"

XML Files and International Characters

XML files, for example message files after internationalization, may contain characters that will need to be translated to NetPoint. It is not possible to provide an exhaustive list of these characters. The general technique is to replace the offending character with its escaped hex equivalent. For example, an accented o, which has the hex equivalent F2, must be shown in the XML file as &#xF2.

DN Validation

At the NetPoint application level, you can optionally control the view and modify functions of the DN type attributes, such as manager or uniquemember, by validating the DN attribute's values. The login user can be allowed to view or modify only those values of the DN for which they have view access (on the class attribute of the objectclass of the DN) as well as localized access; that is, this DN falls under the user's searchbases with respect to the type of objectclass of the DN.

This DN validation is optional and it can be turned on or off through the parameter catalog modifications. Note that the DN validation is an expensive operation, therefore if you do not want so much security, Oblix recommends this validation be turned off. But if security is important, this DN validation must be turned on.

The parameters are in this file:

```
CORElid_install_dir/identity/oblix/apps/common/bin/
oblixappparams.xml
```

and can be overridden by each application.There are two parameters each for view mode and modify mode. For view mode, there is a Boolean parameter called “validateAllDnViewMode”. There is also a parameter in vallist format, called `validateDnAttrsViewMode`. This is used only if the Boolean parameter is false.

The Boolean parameter provides a way to turn the validation on or off *globally*. The Vallist provides the option of turning the validation on/off on an attribute by attribute basis. Thus, the vallist parameters will only be used if the Boolean parameter is false.

Similarly, there are two corresponding parameters for the modify mode called `validateAllDnViewMode` and `validateDnAttrsModifyMode`. By default, in the shipped product, only the Boolean parameters are listed and they are set to false.

Note: The DN validation is always on for IdentityXML calls, for all DN type attributes.

Overriding Windows NT/2000 Default Authentication

The IIS Web server on Windows NT and 2000 supports Challenge/Response Authentication, which defaults to on when IIS is installed. This allows NT users to use their NT domain log-ins when requesting resources from IIS and can conflict with NetPoint's authentication.

For example, on the first request from an Internet Explorer (IE) browser to a resource on IIS protected by NetPoint requiring basic authentication, IE displays a login dialog box requesting a domain along with the user name and password login provided by NetPoint.

To disable Windows Challenge/Response Authentication

1. Run the Microsoft Management Console for IIS.
2. Select the Web Server Host under Internet Information Server in the left hand panel.
3. Right click and select Properties.
4. Scroll down and select Edit the Master Properties for WWW Service.
5. Select the Directory Security tab.
6. Select Edit Anonymous Access and Authentication Control.
7. Complete the step below for your platform:

   NT—Uncheck the Windows NT Challenge/Response checkbox.

   Windows 2000—Uncheck the Integrate Windows Authentication checkbox.
8. Click OK.
9. In the Windows IIS properties screen, click OK.
10. Close the Microsoft Management Console.

Using NetPoint for Authorization Only

In this scenario, you have in place a satisfactory method for authenticating users but would also like to use NetPoint's authorization services.

The solution is to add an authentication scheme to be used in this instance. Using LDAP tools, change the challenge method for this scheme to a special value that the Access Server recognizes for this situation, and instruct the Web server to do the NetPoint authentication/authorization processing after the other authentication method has been applied.

Note: The technique described here is specific to and verified only with iPlanet's Web Server 4.1 and Directory Server 4.11.

The key to this solution is to define an authentication scheme using the challenge method `ext`, which is provided through NetPoint's GUI. This scheme is coupled to the iPlanet variable `auth-user`, which the existing authentication method is expected to set when it authenticates. NetPoint finds this variable already set when it attempts verification, and therefore does not send a challenge back to the browser, but instead goes on to perform authorization.

The following tasks must be completed in order to support NetPoint for authentication only.

Task overview: Implement NetPoint authentication

1. Within the directory, ensure that all user records contain an attribute that can be used by the authentication scheme. The user name attribute, uid, is an example.
2. Within NetPoint System Console, add a new authentication scheme, which will use a special challenge method. See the *NetPoint 7.0 Administration Guide Volume 2* for details.
3. Within the directory server `obj.conf` file, change the processing order to allow the non-NetPoint authentication process to set the authentication result before NetPoint is used. With the authentication check already completed, NetPoint will simply go on to do authorization processing.

All user records in the directory will have an instance of an attribute that will be checked as part of the existing authentication method. Most likely, this will be the user name, `uid`.

To use NetPoint for authorization only

1. Define an *External* Authentication Scheme.

   For example:

   a) Log in to the Access System Console.

b) Go to the Access System Configuration screen, click Authentication Management, and click Add.

c) Create a new authentication scheme, as described in the *NetPoint 7.0 Administration Guide*, and note the following:

The display `Name` and `description` can be any text you choose.

`Level` should be consistent with other authentication schemes you might be using, but can be any value.

For the `Challenge Method`, use `ext`.

For the `Challenge Parameter`, you *must* enter `creds:auth_user`, to match the name of the internal variable carried by iPlanet.

For `SSL Required`, use `No`. This will not be used, since no redirection will take place.

`Required Challenge Redirect` is left blank, for the same reason.

The value for `Plugin(s)` is critical. Enter only one.

- Order - is set to 1.
- `Plugin Name` *must* be `credential_mapping.`
- `Plugin Parameters` are two values separated by a comma. First is the name of the NetPoint mapping base. The second is an LDAP filter specifying the name of the attribute in the directory whose value is expected to match `auth-user`. An example of this entry is:

  ObMappingBase=”o=Company, c=US”, obMappingFilter=”
  (&(objec\]”
  M NB\=-09876UYT5R4E3tclass=inetOrgPerson)
  (uid=%auth-user%))”

2. Change the processing order in the iPlanet `Obj.conf`, to allow the existing authentication method to execute first, and set the iPlanet `user-auth` variable.

For example:

a) In the `obj.conf` file for the Web Server, locate the line:

```
AuthTrans fn="OBWebGate_Authent"
```

b) Comment it out by inserting a “#” at the beginning of the line.

c) Copy the line, without the “#”, to the line directly below the line reading:

```
PathCheck fn="check-acl" acl="default"
```

d) In this new line, change `AuthTrans` to `ObjectType`.

The `obj.conf` file content would then appear as follows:

```
<Object name="default">
#AuthTrans fn="OBWebGate_Authent"
NameTrans fn="pfx2dir" from="/oblix/apps/webgate/bin/
      webgate.cgi" dir="/" name="web_gate"
NameTrans fn="pfx2dir" from="/oberr.cgi" dir="/"
      name="oberr"

. . .

PathCheck fn="check-acl" acl="default"
ObjectType fn="OBWebGate_Authent" dump="true"
```

Note: This change, done in this way, is necessary in order to force WebGate to run after check-acl authentication. Netscape ACL processing, including authentication, is done in the PathCheck check-acl directive. WebGate needs to run after check-acl in order to pick up the auth-user variable whose value is expected to have been set by the authentication method. It will not work to simply put a PathCheck directive for WebGate after the check-acl directive because check-acl is always the last PathCheck directive run, regardless of the order given in `obj.conf`. The method described here, putting WebGate in as an ObjectType directive, has been tested and works.

3. Stop and start the Web Server and the Access Server to ensure that these changes go into effect.

Denying Access to Unprotected Resources Automatically

If you want NetPoint to deny access to all resources that are not explicitly protected by a policy domain, use the WebGate parameter *DenyOnNotProtected*.

NetPoint normally allows access to all resources that are not specifically protected by a policy domain. You can change this behavior using a WebGate parameter.

The WebGateStatic.lst file is installed as part of each WebGate installation:

`$WebGate_install_dir``/access/oblix/apps/webgate/WebGateStatic.lst`

where *WebGate_install_dir* is the directory where WebGate is installed.

This file contains the attribute *DenyOnNotProtected*, which is false by default. Set *DenyOnNotProtected* to true to instruct the WebGate to deny access to any user if the requested resource is not protected by a policy domain.

Note: Be sure to modify this parameter for each installed WebGate individually.

6 Customizing Access Control with Plug-Ins

NetPoint provides APIs that allow software developers to write custom programs or components that integrate closely with NetPoint. These modules may represent anything from custom extensions of base NetPoint functionality to significant applications that are outside of NetPoint, but need to interact with NetPoint for identity or access control functions.

This chapter describes several methods of working with NetPoint programmatically:

Customizing AccessGate/WebGate

Problem: NetPoint provides a standard WebGate component, which is used to control access to a Web server. You want to use NetPoint's access control system to control access to an application server or a function within a standalone application.

Solution: Use the Access Server API, as discussed in the *NetPoint 7.0 Developer Guide*, to create a custom AccessGate.

NetPoint provides a software developer's kit (SDK) that can be used to create an interface to NetPoint's authentication and authorization services. This interface can be built into commercially available application servers, such as BEA's WebLogic, IBM's WebSphere, or iPlanet's Application Server, or any other application that can access the NetPoint Access Server. The application, with the API added, then acts as an *AccessGate* to the Access Server.

In particular, the Access Server API allows Java (servlets, JSPs, EJBs, and so on), C++ (COM/ASP), and C applications to:

- Authenticate users
- Support secured single sign-on (SSO) across Web and application servers
- Authorize user requests for application resources (URLs, EJBs and their methods, and user-defined resources)
- Support protection of non-HTTP resources
- Provide both Java Bean level and Enterprise Java Bean level security

The API is designed primarily to support J2EE-compatible application servers, in particular the way they work with servlets, Java Server Pages, and Enterprise Java Beans, and so is designed from a Java perspective. The API also provides bindings for C++ and C.

Creating an AccessGate is a significant programming task, and for that reason is covered in greater detail in the *NetPoint 7.0 Developer Guide*.

Customizing Authentication Plug-ins

Problem: You want to create an authentication method, for example for a new certificate type, that is not covered completely by the existing plug-ins provided with NetPoint. Or, you want to add a method to authenticate users against an external data store, such as an RDBMS.

Solution: Use the Authentication Plug-in API, as described in the *NetPoint 7.0 Developer Guide*, to write the new plug-in and add it to NetPoint.

When a browser, for example, requests a resource from a NetPoint-protected Web server, the WebGate plug-in checks to see if the resource is protected and if the user needs to authenticate. If so, WebGate requires a new login for the user and sends an authentication challenge to the browser. The challenge conforms to the challenge method defined in an authentication scheme. The authentication scheme in turn is part of an authentication rule which is part of the access policy protecting the resource. When the scheme is carried out, it invokes a single authentication plug-in, or two or more chained plug-ins which are performed in a specified order. The *NetPoint 7.0 Administration Guide Volume 2* provides an introduction to authentication schemes and describes steps for assigning and ordering plug-ins in an authentication scheme.

All schemes follow the same general flow. In response to the authentication challenge, the browser obtains credentials from the user, such as a user name and password or a client certificate. In some cases, for example client certificate authentication, credentials are generated by the browser on behalf of the user. The browser sends the credentials to the server, in a format determined by the challenge. WebGate re-formats the credentials as a set of name-value pairs for use during its processing and treats them as an authentication request.

Input to the single plug-in, or to each plug-in in the scheme, is the set of credentials. Output is a status, to either accept, deny, continue or abort the authentication, and a set of credentials, possibly different from the originals. A result message is logged in the audit file if authentication is denied. When the authentication scheme finishes, the result *must* be to have produced one and only one valid user DN, or, if authentication fails, no user DN.

If authentication succeeds, WebGate creates a session cookie containing the user's profile DN, the IP address of the user's browser, the level of authentication successfully performed, and an expiration timestamp for the cookie. WebGate can also set HTTP header variables based on the authentication actions defined for the authentication scheme. The cookie and HTTP information are returned to the browser, and access is granted.

Creating an authentication plug-in is a significant programming task, and for that reason is covered in greater detail in the *NetPoint 7.0 Developer Guide*.

Customizing Authorization Plug-ins

Problem: NetPoint associates collections of resources into domains, and provides a way for users to set policies controlling access to the domains. You want to add coverage for something other than the NetPoint default resources. For instance, you want to apply an authorization algorithm that is influenced by rules or other data that reside in an external data store, such as an RDBMS.

Solution: Use the Authorization Plug-in API, as discussed in the *NetPoint 7.0 Developer Guide*, to write the new plug-in and add it to NetPoint.

The API provides a way for the user to create functional modules, called plug-ins, which are used within an authorization *scheme*. Schemes are included in authorization rules, and one or more authorization rules, along with one authentication rule and one audit rule, make up a *policy* that controls access to a resource type within a *domain*, such as certain URLs within a Web site or a set of methods within an application. NetPoint provides two standard resource types, URL and EJB, but others can be easily added and defined by the user. See the *NetPoint 7.0 Administration Guide Volume 2* for methods to create resource types, domains, policies, rules and schemes.

Plug-ins within authorization schemes are used for two purposes:

- To confirm or deny access to a resource, or to acquire data to be used by the next authorization rule in the policy. This is called an *authorization plug-in*.
- To perform an action after the access decision is made. This is called a *custom action plug-in*.

To use a plug-in created by the Authorization Plug-in API, two types of information need to be configured by an administrator:

- An authorization scheme to use the plug-in. A given scheme can be used by both authorization plug-ins and custom action plug-ins.
- A custom authorization rule to use the scheme.

Creating an authorization plug-in is a significant programming task, and for that reason is covered in greater detail in the *NetPoint 7.0 Developer Guide*.

Customizing NetPoint to Interact with External Systems

Problem: You want to insert logic that will communicate with an application or perform an action outside of NetPoint.

Solution: Use the Identity Event Plug-in API, as discussed in the *NetPoint 7.0 Developer Guide*, to create the necessary logic and tie it to events that occur within NetPoint.

The Identity Event Plug-in API gives systems integrators the ability to extend the reach of NetPoint beyond its base COREid System and Access System functionality. It does this by providing a channel for COREid System data to flow between NetPoint applications and a wide range of external software components. The potential applications for this API can be as simple as basic logging of NetPoint usage, or as sophisticated as data-filtering pipelines or seamless bridges to ERP systems from third parties like PeopleSoft and Oracle.

The Identity Event Plug-in API is a standard installed component of the NetPoint product.

Creating an Identity Event plug-in is a significant programming task, and for that reason is covered in greater detail in the *NetPoint 7.0 Developer Guide*.

A XML Background

This Appendix provides overviews of XML, XML schemas, and XSLT, for those who may need it in order to follow the discussion and examples for these topics in the main chapters of this guide:

Full descriptions and specifications for this information are available at the following site under XML, XML Schema, and XSL:

www.ww3.org

Find documentation for XML at:

http://www.w3.org/XML/

Find documentation for the XML Schema at:

http://www.w3.org/XML/Schema

A tutorial on the XML schema syntax is available at:

http://www.w3.org./TR/xmlschema-0

Find documentation for XSL and XSLT at the following Web sites:

http://www.w3.org/Style/XSL/

http://www.w3.org/TR/xslt

A good description of template priorities can be found at:

http://www.vbip.com/books/1861003323/chapter_3323_09.asp

XML

XML stands for Extensible Markup Language. It is a set of rules that define *tags* that break a document into parts and identify the parts of the document. These tags define a syntax that can then be used in combination with an XSL stylesheet to reconstruct the document.

The tags that are defined must follow the XML rules, but their content and arrangement can be anything the developer wants. A file of XML text, arranged to represent a certain document, is called an XML application. NetPoint's OutputXML is an XML application, designed to create HTML which will in turn present NetPoint pages to a browser.

NetPoint also uses XML as a structured way to provide some parameters that control its operation. This is a different use than for OutputXML, but since the applications are much shorter and the XML syntax rules are followed here as well, one of these files will serve as an example. For example, frontpageadminparams.xml has the following content:

```
<?xml version="1.0" ?>
<ParamsCtlg xmlns="http://www.oblix.com"
     CtlgName="frontpageadminparams">
   <CompoundList ListName="">
      <SimpleList>
         <NameValPair ParamName="top_frame"
            Value="_top" />
         <NameValPair ParamName="top_main_frame"
              Value="main_frame" />
         <NameValPair ParamName="min_location_area"
            Value="400" />
      </SimpleList>
   </CompoundList>
</ParamsCtlg>
```

This indented presentation, showing the tag levels, is an automatic feature of Microsoft's Internet Explorer. XML editors will also show the file in this way.

Some important parts of this file are the following:

```
<?xml version="1.0" ?>
```

This, the *XML declaration*, is the first line of any well-formed XML application. Internet Explorer and some editors will not show the file as formatted XML unless this line is present. The starting and ending ? make this an *XML processing instruction*. `version="1.0"` is an *attribute*. Attributes are name-value pairs separated by an equals sign, which provide additional information for the instruction. Currently there is only one version of XML.

```
<ParamsCtlg xmlns="http://www.oblix.com"
     CtlgName="frontpageadminparams">
```

<ParamsCtlg is a tag, which starts the definition of the first element in the XML application. The definition ends with the matching closing tag, which has the same form except it uses a / before the tag name:

```
</ParamsCtlg>
```

Everything between the starting and ending tags defines the element `ParamsCtlg.` Nested within it is the element `CompoundList`, which has elements nested within it, and so on. An important attribute is `xmlns`, which stands for *XML namespace*. This specifies an owner and possible reference source for this XML application. We identify ourselves as creators of this application.

```
<NameValPair ParamName="top_frame" Value="_top" />
```

The technically precise way to write this element would have been

```
<NameValPair>
   ParamName="top_frame" Value="_top"
</NameValPair>
```

However, when the definition is a short one like this, the XML rules allow use of an abbreviated closing tag. /> indicates the closing tag for the immediately preceding start tag.

The attributes ParamName="top_frame" and Value="_top" provide the useful content of the file, which is the name of a variable used by NetPoint and its value.

An important concept, essential to the application of stylesheets, is a node. A node is a level within the XML application, described by stringing together the elements that locate it uniquely within the nested elements. For example, ParamsCtlg is the root node for the application. The root node is the element name immediately following the XML processing instruction(s); all other elements are nested within it. Other examples of nodes are ParamsCtlg/CompoundList and ParamsCtlg/CompoundList/SimpleList.

XML Schema

An XML Schema shows and describes the content of an XML application. The following list interprets some of the elements that appear within a NetPoint Schema definition file, based on the first few characters of each element. This is *not* intended to be an explanation of the full XML Schema syntax; see the referenced site for that.

xsd:attribute—Appears within the body of an element being defined, and defines an attribute that belongs to it. Parts of the definition usually present are:

- `name=″xxxx″`—The name of the attribute
- `type=″yyyy″`—The data type for the attribute; see the list below

- `use="required"`—This is present only if the attribute must be present in the output.
- `value="zzzz"`—This is present only if the attribute takes a fixed value.

xsd:choice—Precedes a list of other elements, indicating that one and only one of those elements is allowed. The choice itself can be made from zero to many times, as controlled by the values of minOccurs and maxOccurs. The value of minOccurs is the fewest number of times this element can appear in the list. If the value is zero, the element is optional in the list. The value of maxOccurs is the greatest number of times the element can appear in the list. A value of Unbounded means there is no limit.

xsd:complexType—Most often used in the body of an element that is being defined, and means that the element will contain other elements.

xsd:element name="xxxx"—Declares and within its body goes on to fully define a category of information describing the element xxxx. Most instances of this in the schema files go on to provide a body for the element and build it up from subelements. A few, for example ObTextMesage in the displaytype.xsd file, have no body, in which case they use type to immediately specify the data type of the element.

xsd:element ref="xxxx"—Most often used to provide the name of a subelement for inclusion in a list that is part of the body defining an element. The referenced element will have been defined elsewhere. The element may also include the attributes minOccurs and maxOccurs.

xsd:enumeration—Provides a list of possible values.

xsd:include schemalocation="xxxx"—An element that specifies a file which contains additional XML schema information, to be treated just as if it were provided inline in the current file.

xsd:restriction base="xxxx"—Defines the pattern for values that are used for a data type being defined; see xsd:simpletype. NetPoint uses the restriction base NMTOKEN, which means the value must be a legal XML string and contain no white spaces.

xsd:sequence—Precedes a list of subelements within another element, and indicates that, if they are present, they will appear in the order listed.

xsd:simpletype—This begins the definition of a data type, usually followed by an `xsd:restriction` definition.

Some possible data types are:

xsd:boolean—Acceptable values are true/false, or 1/0.

xsd:date—Acceptable values are dates in the form YYYY-MM-DD (many other date types are possible).

xsd:decimal—Acceptable values are decimal numbers (other number types are possible)

xsd:string—Acceptable values are a string of characters

xsd:time—Acceptable values are a time of day in the form hh:mm:ss.sss.

xsd:uri-reference—Acceptable values are URLs.

All of the Oblix XML schemas are defined within a root element called oblix. The table below shows the schema for the usc_profile.xsd definition of oblix, beginning with its initial definition in component_profile.xsd. The table shows the schema only to the first two node levels below oblix; the full schema goes much deeper. If you look at just the pure OutputXML provided by NetPoint for the view (My Identity) program, this information, in this order, is what you see.

Level 1	Level 2
ObProfile (defined in component_profile.xsd)	ObPanel
	ObHeaderPanel
	ObRequestInfo
	ObScripts
	ObForm
	ObDisplay
	ObTextMessage
	ObButton
	ObStatus
ObNavBar (defined in navbar.xsd)	ObRequestInfo
	ObScripts
	ObMisc
	ObApps
	ObApplication
	ObFunctionsButtons
	ObStatus
ObSearchForm (defined in searchform.xsd)	ObHelpContext
	ObRequestInfo
	ObScripts
	ObForm
	ObDisplay
	ObButton
	ObAdvancedSearch
	ObSearchRow
	ObStatus

Level 1	Level 2
ObApplicationFunc (defined in navbar.xsd)	ObFunctions
	ObRequestInfo
	ObStatus
ObStatus (defined in component_basic.xsd)	This is a string of type xsd:string; it contains no other elements.

XSL and XSLT

XSL stands for Extensible Style Language. Files written in this language are used along with XSLT to create documents. The XSL file itself is a well-formed XML document. The language relies heavily upon the use of *templates*, which are sets of instructions to the XSL transformer, telling it what to produce as output for a particular node within the XML.

XSLT stands for XSL Transformation. This is a process that combines an XML application with an XSL stylesheet to create a document.

General Syntax

The following list interprets some of the elements that appear within a NetPoint stylesheet file, based on the first few characters of each element. This is *not* intended to be an explanation of the full XSL syntax; see the referenced site for that.

Note: In the NetPoint XSL files, lines starting with <xsl: are instructions to the XSL transformer. All others are HTML text to be written verbatim into the HTML output.

---

xsl:apply-templates select="xxxx"—Once the transformer is positioned, using xsl:template-match, to a node within the XML, this element identifies which subnodes or sub-subnodes are to be processed. Point at sub-subnodes within the selected node by providing their nested structure, for example xsl:apply-templates="*xxxx*/*yyyy*", where *yyyy* is a node nested within *xxxx*. If the select option is omitted, templates for all the subnodes under the matched node are processed.

The transformer decides which templates to use by identifying each subnode by name, and then searching the entire stylesheet for the best xsl:template match for that name. The match will generally be on the last node in the nested list, for example *yyyy* in the above example. The instructions for that matched node are applied immediately.

xsl:attribute name=”string”—Inserts the text specified by *string* into the output.

xsl:call-template name=”xxxx”—Immediately performs the transformation required by the template *xxxx*. The template to be called will have been specified using xsl:template name=”*xxxx*”.

xsl:choose—Precedes a list of possible transformations, each of which is indicated by the use of the xsl:when element. It may be that none of the xsl:when elements applies; the xsl:otherwise element covers this possibility. If more than one of the xsl:when elements is true, only the first true xsl:when element is applied.

xsl:for-each select=”xxxx”—Applies the content of this element to all occurrences of *xxxx*.

xsl:if test=”expression”—Allows a choice to be made. If expression evaluates to a Boolean true, the content of the xsl:if element is performed. If not, it’s not performed. Expression syntax is described below.

xsl:include href =”xxxx”—An element that specifies a file which contains additional XSL stylesheet information, to be treated just as if it were provided inline in the current file.

xsl:number value=”expression”—Used to insert a formatted integer into the output. In Oblix stylesheets, expression often uses the position() function, which indicates the position of a node in a list, starting with 1.

xsl:otherwise—The last element in the list of elements under an xsl:choose, following the xsl:whens, which is to be applied if none of the xsl:whens is true.

xsl:template match=”xxxx”—Point the transformer to the node named xxxx in the XML data. Point at subnodes by providing their nested structure, for example xsl:template-match=”*xxxx/yyyy*”, where *yyyy* is a node nested within *xxxx*. This must be followed by one or more uses of xsl:apply-templates, otherwise no transformation of the XML data will be done.

xsl:template name=”xxxx”—Create a named template, to be applied when xsl:call-template=”*xxxx*” is used.

xsl:value-of select=”expression”—Inserts the value specified by *expression* into the output.

xsl:when test=”expression”—Allows a choice to be made. If *expression* evaluates to a Boolean true, the content of the xsl:when element is performed. If not, it’s not performed. Usually, multiple xsl:when elements are nested under an xsl:choose element.

Expression Syntax

Again, this is only a subset of a much longer list, provided to allow you to interpret NetPoint XSL files. Expressions can be of several kinds:

- **Node Sets**—A node set describes a set of nested elements, in the form *xxxx*/*yyyy*/*zzzz*, meaning the element *zzzz* is nested within the element *yyyy* which is then nested within the element *xxxx*. When a node set is used as the expression for a test, the test is true if the nested set exists in the XML, false if it does not.

  Further, this may be used in the form *xxxx*/*yyyy*/*zzzz*[@attribute = a value]. This means to look at the value of the attribute belonging to element *zzzz*. The expression is true if the attribute has the specified value and false otherwise.

- **String Content**—One form of this is

  ```
  <xsl:value-of select="@attribute" />
  ```

  which means return the value of the attribute.

  Another is

  ```
  <xsl:if test="@attribute">
  ```

  which is true if the attribute is valid for the element and has a non-`NULL` value.

- **Numeric Content**—In this case, the expression reduces to a number. An example is

  ```
  <xsl:number value="position()-1">
  ```

  which gives a number one less than the position of the current element in a list of elements.

Client-Side Transformation

Client-side processing of stylesheets is supported only with Microsoft Internet Explorer (IE) 5.5 and later. Earlier versions of IE require installation of a patch.

To set this up

1. Install the latest msxml patch.

   This must be msxml3.0 (or above), which can be obtained from:

   ```
   http://download.microsoft.com/downloads/
   xml/Install/3.0/WIN98Me/EN-US/msxml3.exe
   ```

2. Install the registration tool for `msxml`.

   This can be obtained from:

   ```
   http://msdn.microsoft.com/msdn-files/027/
   001/469/xmlinst.exe
   ```

3. Enter the following command sequence:

```
xmlinst -u
regsvr32 -u msxml.dll
regsvr32 msxml3.dll
xmlinst
```

4. Change the controlling parameter.

   In the *$COREid_install_dir*/identity/oblix/apps/common/bin/globalparams.xml parameter file, change the value for OutputFormat from default to xml.

5. Restart the COREid Server.

6. Verify the change.

   To verify that this change indeed took place, enter the COREid System using an Internet Explorer 5 browser. If you do a view source, you will see XML instead of HTML.

NetPoint XSL Transformation Limits

NetPoint has a built-in XSL Transformation processor. This processor implements most, but not all, of the XSLT standard. The following is some information applying to the NetPoint version.

- The processor does not insert the declaration line:

  ```
  <?xml version="1.0" ?>
  ```

  in XML files that it generates. If this is needed because you want to see an indented XML presentation, you must include it in the stylesheet.

- The processor does not support UTF characters in a sort. An attempt to do this will generate an error report.

- The processor has a stack limit depth of 5298; recursive templates can go no deeper than this.

- The processor assumes that its output is intended for use by a browser and formats output with an HTML formatter.

- The processor is intended primarily for use in a production environment, where performance is important. It does only minimal checking of stylesheet syntax. Very bad syntax can crash the processor. Only known stylesheets with validated content should be used in the production environment. Validation tools are listed in “XSL Validation” on page 190.

- Embedded stylesheets in the XML are not supported.

- Full support, or in some cases, any support, of the following commands is not provided. If you need to use these commands, double-check your results before putting the stylesheet into production.

- XSL:format-number
- XSL:output
- XSL:document
- XSL:namespace
- XSL:comment
- XSL:format
- XSL:processing instruction
- XSL:sort—case order
- XSL: id

For more information, see “Useful Tools” on page 183.

B Useful Tools

Tools are available for you to use to make changes to the installed NetPoint files, such as parameter files and directories. This chapter discusses these tools. Topics include:

Text Editor

NetPoint operation is influenced by the content of various ASCII text files, in particular parameter files, also called *.xml files. These can be edited with a text editor.

Here are some guidelines to observe when editing these text files:

- Always make and save a backup copy of the original file. This provides a way to recover if you make a mistake.
- Do not insert blank lines.
- Do not insert comments in the file.
- Remember to stop and restart the COREid Server to force to force reloading of the changed data, whose original values may have been cached. For certain changes, you will also have to restart your Web server.
- Verify that the intended change has taken effect.
- Make one change at a time if possible, to make it easy to find any mistakes. Whether or not you do this, keep a record of what you changed and when, so that if users start to report problems you can quickly find out if they relate to your configuration changes.

- Use a simple text editor, such as *NotePad* on NT or *vi* on UNIX, to avoid introducing non-ASCII characters into the file. You can also use an XML editor if you have one to reduce the chance of introducing errors when editing XML files. Some XML editors will check the file for well-formedness for you.

LDAP Tools

Directory applications use Lightweight Directory Access Protocol (LDAP) as a standard tool to create, modify and report data stored within the directory. Specific tools are available to allow relatively easy manipulation of this data directly, using LDAP.

This section provides a short introduction of these tools and methods for using them. More detail is available from the manufacturer of your server application, specifically for its version of these tools.

Viewing Directory Content in LDIF Files

The structure of a directory, and the data contained within it, is represented by the content of an LDAP Data Interchange Format (LDIF) file. The file can be *output*, the formatted result of a request made to the directory by an LDAP reporting tool, such as LDAPSEARCH. It can also be *input*, data that is intended for insertion to the directory, either as entirely new data, or as an update to existing data, using an updating tool such as LDAPMODIFY.

The following is an example, part of an LDIF file taken from a NetPoint Demo Directory:

```
dn: cn=John Kramer, ou=Sales, o=Company, c=US
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: companyOrgPerson
cn: John Kramer
sn: Kramer
telephonenumber: 415-555-5269
facsimiletelephonenumber: 415-333-1005
title: Account Manager
departmentnumber: 1204
employeetype: Fulltime
employeenumber: 521-321-4560
givenname: John
.
.
```

Reporting Directory Content with LDAPSEARCH

LDAPSEARCH is one possible tool that can be used to report directory content. There are others, which use a different syntax, but the concepts are the same.

LDAPSEARCH can be used in either a command line or interactive mode. The command line approach is preferable, as it allows users to provide the text of the report request by means of a input file. It is easy to verify the content of this file before making the request. Errors are corrected by changing a few characters in the file rather than retyping the full request, which would be necessary in the interactive mode.

LDAPSEARCH Command Line Format

The command line format for LDAPSEARCH is:

```
ldapsearch <params> <filter> <attr_list>
```

Note: Each of the three categories shown between <> is optional; if all are omitted, LDAPSEARCH drops into interactive mode, which is not discussed here.

The categories are as follows:

- ***params***—These *parameters* tell LDAPSEARCH how to operate. One of them, `-f`, is used to specify a filter file. If instead the search filter is provided on the command line, all parameters must be stated before the filter is stated.
- ***filter***—The *filter* instructs LDAPSEARCH to provide a subset of the data that would otherwise be provided. For example, a filter could require that only names beginning with N be reported. A filter provided on the command line must be enclosed within quotes.
- ***attr_list***—The *attribute list*, if included on the command line, overrides the default attribute listing. The default list shows all attributes belonging to the directory entry, except operational attributes. If you wish to see only some of these attributes listed, provide their names in the command line, following the filter and separated by spaces. If you want to see operational attributes, provide their names in the command line. If you follow the operational attributes with a `*` you get the default list of attributes as well.

LDAPSEARCH Command Line Parameters

Parameters are always provided in the form:

```
-p pdata
```

where `p` is the parameter, preceded by a dash, and `pdata` is the information required for the parameter, if any. If the data contains a space, it must be completely enclosed in double quotes:

```
-p “pdata with spaces”
```

Following is an alphabetical list of commonly used parameters. There are others. See the reference document for your version of LDAPSEARCH, or use the parameter `/?` to see them listed.

-A—Tells the search to retrieve the attribute names only, not the attribute values.

-b—Searchbase, the starting point for the search. The value specified here must be a distinguished name that is in the directory. Data provided for this parameter MUST be in double quotation marks. For example:

```
-b “cn=Barbara Jensen, ou=Development, o=Oblix.com”
```

-D—Distinguished name of the server administrator, or other user authorized to search for entries. This parameter is optional if anonymous access is supported by your server. For example:

```
-D “uid=j.smith, o=Oblix.com”
```

-f—Specifies the file containing the search filter(s) to be used in the search. For example:

```
-f filterfile
```

-h—Hostname or IP address of the machine on which the directory server is installed. This entry is optional; if no hostname is provided, LDAPSEARCH uses the localhost. For example:

```
-h myserver.com
```

-H—This generates a list of all possible LDAPSEARCH parameters.

-p—Port number that the directory server listens at. For example:

```
-p 1049
```

-s—Scope of the search. The data provided for the scope must be one of the following:

base—Search only the entry specified in the -b option.

one—Search only the immediate children of the entry specified in the -b parameter; do not search the actual entry specified in the -b parameter.

sub—Search the entry specified in the -b parameter, and all of its descendants. That is, perform a subtree search starting at the point identified in the -b parameter. This is the default, if the -s parameter is not used.

-S—Designates the attribute(s) to use as sort criteria, controlling the order in which the results are displayed. You can use multiple -S arguments if you want to sort on multiple criteria. The default behavior is not to sort the returned entries. In the following example, the search results are sorted first by surname and then, within surname, by first name:

```
-S sn -S firstname
```

-w—Password associated with the distinguished name that is specified in the -D option. If you do not specify this parameter, anonymous access is used. For example:

```
-w password
```

-x—Specifies that the search results are sorted on the server rather than on the client. This is useful if you want to sort according to a matching rule, as with an international search. In general, it is faster to sort on the server than on the client.

-z—Specifies the maximum number of entries to return in response to a search request. For example:

```
-z 1000
```

Examples

If you wanted to get the surname (sn), common name (cn), and given name for every employee within the sales organization whose given name is John, from the directory server listening at port 392, entirely from the command line, you could provide the following information:

```
ldapsearch -p 392 -b “ou=sales, o=company, c=US” -s sub “givenname=John”
sn cn givenname
```

Results could be something like:

```
dn: cn=John Jackson, ou=Sales, o=Company, c=US
sn: Jackson
cn: John Jackson
givenname: John
dn: cn=John Kramer, ou=Sales, o=Company, c=US
sn: Kramer
cn: John Kramer
givenname: John
```

You can get the same results by using a filter file. For example, a file called `namejohn` containing the filter:

```
givenname=John
```

can be used, with the command line being the following:

```
ldapsearch -p 392 -b “ou=sales, o=company, c=US” -s sub -f namejohn sn
cn givenname
```

Changing Directory Content with LDAPMODIFY

LDAPMODIFY is a tool that can be used to change or add directory content. There are others, but the concepts are similar.

LDAPMODIFY opens a connection to the specified server, using the distinguished name and password you supply, and modifies the entries based on LDIF update statements contained in a specified file. LDAPMODIFY can also be run in interactive mode, a method that is not discussed here.

If schema checking is active when you use LDAPMODIFY, the server performs schema checking for the entire entry before applying it to the directory. If the directory server detects an attribute or object class in the entry that is not known to the schema, then the entire modify operation fails. Also, if a value for a required attribute is not present, the modify operation fails. Failure occurs even if the value for the problem object class or attribute is not being modified.

Note: Turn on schema checking at all times, as a matter of good practice, to avoid accidently adding data to the directory that could later be unusable or cause schema violations when schema checking is turned back on. Schema checking is controlled at the directory administration server console and is generally on by default.

LDAPMODIFY Command Line Format

The command line format for LDAPMODIFY is:

```
ldapmodify <params>
```

Note: The params category is optional; if it is omitted, LDAPMODIFY drops into interactive mode, which will not be discussed here.

---

where `<params>` are *parameters* that tell LDAPMODIFY how to operate. One of them, `-f`, can be used to specify a file describing modifications to be made to the directory.

LDAPMODIFY Command Line Parameters

Parameters are always provided in the form:

```
-p pdata
```

where `p` is the parameter, preceded by a dash and followed by a space, and `pdata` is the information required for the parameter, if any. If the data contains a space, it must be completely enclosed in double quotes:

```
-p “pdata with spaces”
```

Following is an alphabetical list of commonly used parameters. There are others. Use the parameter `/?` to see them listed.

-a—Allows you to add LDIF entries to the directory without requiring the `changetype:add` LDIF update statement, which is necessary in the interactive mode. This provides a simplified method of adding entries to the directory; in particular, this allows you to directly add a file created by LDAPSEARCH and modified to make changes.

-c—Forces the utility to run in continuous operation mode. Errors are reported, but the utility continues with modifications. Default is to quit after reporting an error.

-D—The distinguished name of the server administrator or other user authorized to change directory entries. This parameter is optional if anonymous access is supported by your server. For example:

```
-D “uid=j.smith, o=Oblix.com”
```

-f—Provides the name of the file containing the LDIF update statements used to define the directory modifications. For example:

```
-f changestomake.txt
```

-h—Hostname or IP address of the machine on which the directory server is installed. This entry is optional; if no hostname is provided, LDAPSEARCH uses the localhost. For example:

```
-h mozilla
```

-H—Lists all possible LDAPMODIFY parameters.

-p—Port number that the directory server uses. For example:

```
-p 1049
```

-w—Password associated with the distinguished name that is specified in the -D option. If you do not specify this parameter, anonymous access is used. For example:

```
-w password
```

Examples

Suppose you want to change the stored given name of John Kramer, as reported under the discussion of LDAPSEARCH. The data reported back was:

```
dn: cn=John Kramer, ou=Sales, o=Company, c=US
sn: Kramer
cn: John Kramer
givenname: John
```

This output can be used to derive an input file, `ToHarvey`, whose content might be:

```
dn: cn=John Kramer, ou=Sales, o=Company, c=US
changetype:modify
replace:givenname
givenname: Harvey
```

The command line would then be:

```
ldapmodify - p 392 -f ToHarvey
```

If you were to now search the directory with the command line:

```
ldapsearch -p 392 -b “ou=sales, o=company, c=US” -s sub
“givenname=Harvey” sn cn givenname
```

The response would be:

```
dn: cn=John Kramer, ou=Sales, o=Company, c=US
sn: Kramer
cn: John Kramer
givenname: Harvey
```

XML/XSL Editors

The following are some links to editors that might be useful in working with XML and XSL files:

```
http://www.w3.org/Style/XSL/
```

(under the XSL-Enabled Authoring Tools)

```
http://www.xml.com/pub/rg/XML_Editors
http://www.xmlspy.com/
```

XSL Validation

Oblix recommends you thoroughly validate XSL stylesheets before using them in production. The following are freeware validators:

- Xalan (testxslt)

  ```
  http://xml.apache.org/xalan-c/index.html
  ```

  This contains a validator called testxslt, version 1.1

- Saxon

```
http://users.iclway.co.uk/mhkay/saxon/
```

- XT

```
http://www.jclark.com/xml/xt.html
```

All three tools will parse and check for syntax errors. The second two do the better job of listing and locating syntax errors, while the first most closely approximates NetPoint's XSLT processor.

Troubleshooting Example

The NetPoint COREid System uses the Xalan/Xerces XSLT processor. The Xalan processor is orders of magnitude faster than the built-in XSLT engine.

Suppose you add templates derived from basic.xsl to a custom xslStyleFunctions.xsl and both are included in almost any customized stylesheet). In this case, Xalan may report a number of problems, including an inability to resolve any of the oblix:ObScript/oblix:ObValue/. This results in empty JavaScript includes, for example:

```
<script=“JavaScript” xml:space=“preserve”\
</script>
<script=“JavaScript” xml:space=“preserve”\
</script>
...
<script=“JavaScript” xml:space=“preserve”\
</script>
```

Note: The src=“...” parameter is missing.

The XSLT standard does not impose requirements on how processors should handle template conflict resolution. As a result, the XMLSPY internal processor does not report errors. However, Xalan would and COREid returns a stylesheet processing error.

In this case, there was a conflict due to multiple same-name templates sharing the same priority. A good description of template priorities can be found at:

```
http://www.vbip.com/books/1861003323/chapter_3323_09.asp
```

Default priorities are assigned by the processor based on certain criteria. Some processors apply smart logic to determine which conflicting template wins, for example, the template closest to the source stylesheet (wf_create.xsl or usc_profile.xsl).

Adopting the following style would protect against template conflict resolution:

- Add a priority of ‘1’ to any templates in basic.xsl.
- Add a priority of ‘2’ to any xslStyleFunctions.xsl templates with the potential for a name conflict.
- Add a priority of ‘3’ to any source (bottom-level) xsl in case there is another override to a template defined in included xsl-s.

The following two procedures prepare you to troubleshoot Xalan-specific stylesheet processing errors reported by COREid within XMLSPY.

To use XMLSPY with Xalan/Xerces on Windows 2000 and above

1. Download a Xalan-C 1.7.0 binary distribution, which is *not* compatible with Xerces-C 2.5.0 distribution.
2. Download a Xerces-C 2.4.0 binary distribution.
3. Unzip the downloaded archives.

   For example:

   C:\opt
4. Navigate to your system PATH variable.

   For example:

   Control Panel > System > Advanced > Env Vars > System Variables > Path
5. Append the following to your system PATH variable:

   C:\opt\xml-xalan\c\Build\Win32\VC6\Release;C:\opt\xerces-c2_4_0-windows_nt-msvc_60\bin;
6. Configure XMLSPY to use an external XSL processor, as discussed next.

To configure XMLSPY to use an external XSL processor

1. Select Tools > Options > XSL.
2. Restart XMLSPY to pick up the newly updated PATH.
3. Select External XSL transformation program.
4. In the text box, enter Xalan.exe -o %2 %1 %3
5. Click OK.

C NetPoint Parameter Files

NetPoint provides a simple means for users to modify the way it operates, by changing the content of specified parameter files, also called catalog files. This appendix describes the file format, provides a list of the files, and describes values within them that you can change to customize NetPoint system operation.

File Categories and Locations

All of the parameter files are located relative to the COREid System installation directory, which could be, for example:

c:\NetPoint\identity\oblix.

on Windows NT, or

/var/NetPoint/identity/oblix.

on Unix.

Note: The remainder of this discussion will refer to paths relative to the COREid System installation directory, and will use the path separator `/`. This is to aid readability; it also happens to be the correct syntax for UNIX systems and URLs, as well as relative paths for external references within XML and other files. When referring to file paths on disk, Windows NT users should replace `/` with `\` as necessary.

The parameter files can be viewed as belonging to one of several categories, distinguished by the type of parameters they contain:

- Parameters that affect the administrative applications: *User Manager Admin; Group Manager Admin; Organization Manager Admin; System Admin.*
- Parameters that affect the user applications: *User Manager; Group Manager; Organization Manager; Asynch Mailer; Password Management; Query Builder; Selector.*
- Parameters whose effect is common across many applications: the user applications, the administrative applications and the *Comm Server* (a binary streaming data module).

- Parameters that affect NetPoint's interaction with the directory database (DB), further subcategorized as follows: *user, group, organization, application, configuration, workflow, and LDAP referential integrity*.
- Parameters that affect NetPoint's Multi-tier Architecture (for example, the webpass Web application, or the COREid Server engine).

The following tables list the names and locations of files containing parameters in each category listed above:

Table 1 Parameter Files for Administrative Parameters

apps/admin/bin/objservcenteradminparams.xml
apps/admin/bin/frontpageadminparams.xml

Table 2 Parameter Files for User Parameters

apps/userservcenter/bin/userservcenterparams.xml
apps/userservcenter/bin/usc_wf_params.xml
apps/groupservcenter/bin/groupservcenterparams.xml
apps/groupservcenter/bin/gscaclparams.xml
apps/groupservcenter/bin/gsc_wf_params.xml
apps/objservcenter/bin/objservcenterparams.xml
apps/objservcenter/bin/osc_wf_params.xml
apps/asynch/bin/asynchparams.xml
apps/querybuilder/bin/querybuilderparams.xml
apps/selector/bin/selectorparams.xml

Table 3 Parameter Files for Common Parameters

apps/common/bin/globalparams.xml
apps/common/bin/oblixadminparams.xml
apps/common/bin/oblixappparams.xml
apps/common/bin/oblixbaseparams.xml
apps/common/bin/comm_serverparams.xml

Table 4 Parameter Files for Directory Interaction Parameters

data/common/appdbparams.xml
data/common/configdbparams.xml
data/common/userdbparams.xml
data/common/groupdbparams.xml
data/common/objectdbparams.xml
data/common/workflowdbparams.xml
data/common/ldapappdbparams.xml

Table 4 Parameter Files for Directory Interaction Parameters

data/common/ldapconfigdbparams.xml
data/common/basedbparams.xml
db/ldap/ldapreferentialintegrityparams.xml

Table 5 Parameter Files for NetPoint Multi-tier Architecture Parameters

apps/webpass/bin/webpass.xml

Procedures for Modifying Parameter Files

The parameter files are read once, when the COREid System starts up. You may modify the files in-place using a text editor or an XML editor, but the changes will not take effect until the next time the COREid Server starts up.

If you have many edits to make, it is always a good idea to make a backup copy of all the files you are going to edit before you begin, so that you have a known state to roll back to if you make a mistake.

The parameter files are not validated by NetPoint; if you are seeing unexpected behavior after making changes, check the COREid System log files located under *COREid_install_dir*/identity/oblix/logs for error messages that might help you locate the problem. A common mistake with XML files when editing with a text editor is to break the XML syntax, for instance by omitting a closing tag. You can check for well-formedness, or avoid such mistakes altogether, by using an XML editor instead.

If more than one COREid Server is installed, a set of catalog files will have been installed under the *COREid_install_dir* of each COREid Server instance. If you want your changes to affect all installed COREid Servers, you must propagate your changes to all instances.

Precedence Rules

Some parameters may exist in more than one file. When this occurs, NetPoint resolves the value using the following heuristics. In all cases, the search stops as soon as the parameter is found:

1. User Application Parameters

   The application-specific parameter file (under the application directory for User Manager, Group Manager, etc.), is searched first.

   Then, the oblixappparams.xml file is searched.

   Then, the oblixbaseparams.xml file is searched.

2. Admin Application Parameters

   The set of application-specific administration parameter files (User Manager Admin, Group Manager Admin, etc.) are searched first.

   Then, the oblixadminparams.xml file is searched.

   Then, the oblixbaseparams.xml file is searched.

3. Directory (DB) Parameters

   The set of parameter files specific to the DB (ldapuserdbparams, etc.) are searched first.

   Then, the default DB parameter files (userdbparams.xml, appdbparams.xml, etc.) are searched.

   Then, the basedbparams.xml file is searched.

Parameter File Format

Parameter files are expressed in XML. They have a simple structure, and make extensive use of user-friendly names to aid in working with the files.

When working with parameter files, it is essential that you limit your changes to only the text falling within quotation marks and strictly follow the rules for each kind of change.

The following excerpt is from the userservcenterparams.xml file. Methods for providing the parameter values are highlighted in bold and discussed below.

```
<?xml version="1.0" ?>
<ParamsCtlg xmlns="http://www.oblix.com"
     CtlgName="userservcenterparams">
  <CompoundList ListName="">
    <SimpleList>
      <NameValPair ParamName="top_frame"
           Value="_top" />
      <NameValPair ParamName="top_main_frame"
           Value="main_frame" />
      <NameValPair ParamName="min_location_area"
```

```
            Value="400" />
      </SimpleList>
      <ValList ListName="search_result_views">
         <ValListMember Value="table_view" />
         <ValListMember Value="custom_view" />
      </ValList>
      <SimpleList>
         <NameValPair ParamName="ObEnhanceSearch"
            Value="true" />
      </SimpleList>
      <ValNameList ListName="ObEnhanceSearchList">
         <NameValPair ParamName="OOS"
               Value="That Contains" />
      ...
      ...
         <NameValPair ParamName="OSL"
               Value="That Sounds Like" />
      </ValNameList>
      <SimpleList>
         <NameValPair ParamName="navbar_bgcolor"
               Value="#669966" />
      </SimpleList>
   </CompoundList>
</ParamsCtlg>
```

There are three methods of providing parameter values. These are shown in bold in the excerpt above:

a) `<SimpleList>`

The `SimpleList` element provides a simple list of `NameValPair` elements giving parameter names and their values. The parameter names (`ParamName`) are known to the COREid Server Manager and are expected to be present. The parameter names and legal values, for this and the other methods, are provided under “Parameter Reference” on page 199.

b) `<ValList ListName="search_result_views">`

The ValList element provides a list of options, such as methods of execution or a choice of display format, as a set of ValListMember elements that are available to the COREid System. The name of the method or format goes in the value attribute. These names are predefined and cannot be changed. You can enhance flexibility for the COREid System by adding a new ValListMember entry. You can reduce functionality by removing a ValListMember element. For example, if you remove the line

```
<ValListMember Value="custom_view"/>
```

the COREid System is no longer able to display a custom view.

For this type of change, the *Parameter Name* column in the tables that follow actually shows the `ListName`.

c) `<ValNameList ListName="ObEnhanceSearchList">`

The ValNameList element is similar to the SimpleList element, because it provides a list of NameValPair elements. NetPoint generally uses ValNameList parameters to construct pull-down menus in the GUI. The list includes a parameter name (ParamName) and a value for the text describing it. The parameter names are predefined and cannot be changed. You may add them to the list, remove them from the list, or change the text displayed for the parameter in the GUI pull-down menu by changing the content of the value attribute.

For example, if you remove the line

```
<NameValPair ParamName="OOS" Value="That Contains" />
```

OOS will no longer appear as a search option. If instead you change the line to the following

```
<NameValPair ParamName="OOS" Value="That Holds" />
```

OOS will be described as “That Holds” in the GUI pull-down menu.

For this type of change, the *Parameter Name* column in the tables shows the ListName.

Parameter Reference

The following tables describe the parameters that may be present in each parameter file.

The key to the table columns is as follows:

Parameter Name—The name of the parameter. In some cases, a parameter takes a set of subordinate parameters, whose names are listed.

Description—What the parameter is used for.

Default Value—The factory default value in the file when installed.

Possible Values—Alternative values that you can enter for the parameter.

Table 6 userservcenterparams.xml

Parameter Name	Description	Default Value	Possible Values
`min_location_area`	The area allocated for the location GIF. This depends on each customer's location image.	400	A positive integer
`navbar_bgcolor`	The background color for the application navigation bar. The value is presented in the obbgcolor attribute of the ObNavbar element.	#669966	Any RGB value
`ObEnhanceSearch`	Enables extended search UI and functionality.	true	True or false
`ObEnhanceSearchList`	If the ObEnhanceSearch parameter is set to true, the search page is enhanced by adding a drop-down list of search operators. This list is constructed using the ObEnhanceSearchList parameter. The list contains a set of NameValPair elements. The following ParamName (Value) attribute pairs are supported for all applications: OOS (That Contains) OSM (Contains in Order) OEM (=) OLE (<=) OGE (>=) OBW (That Begins With) OEW (That Ends With) OSL (That Sounds Like) The Value text in parentheses above describes the semantics of each value, and is also the default text displayed to the user in the drop-down list (you can change this in the catalog). The ParamName, *Oxx*, is not displayed; it is an operation code sent to the application doing the search.	OOS,OSM,OEM,OLE,OGE,OBW,OEW,OSL	All applications: OOS,OSM,OEM,OLE,OGE,OBW,OEW,OSL
`search_result_views`	Display format for User Manager search results. User Manager supports Table Format and Custom Format.	table_view custom_view	table_view custom_view

Table 6 userservcenterparams.xml

Parameter Name	Description	Default Value	Possible Values
searchString MinimumLength	The minimum number of characters that the user must provide in order to perform a search operation. NOTE: This parameter does not appear in the installed version of this file. If added by the user, the value here applies only to User Manager.	3	Any positive, non-zero integer
top_frame	Name of the top browser frame User Manager.	_top	A frame name
top_main_frame	Name of the main browser frame in User Manager.	main_frame	A frame name

Table 7 groupservcenterparams.xml

Parameter name	Description	Default Value	Possible Values
groupMember SearchString MiminumLength	This specifies the minimum length of the search string that the user must specify in order to do a member search. This is used only in the Group Manager View Members page, where the user can search for members using specific search criteria. If this parameter's value is 0, then the user is allowed to do a blank search, in which case the application will display all the members of this group. If this parameter has any other value, then the user can only do a search if the search string has at least that many characters.	3	Any non-negative integer, including zero
`navbar_bgcolor`	The background color for application navigation bar. The value is presented in the obbgcolor attribute of the ObNavbar element.	#9999CC	Any RGB value
`ObEnhanceSearchList`	Drop-down selection of Search Condition in Search toolbar. The Name is a Search Condition understood by the application. The value is a display name on the selection menu.	OOS:That Contains OSM:Contains In Order OEM:= OLE:<= OGE:>= OBW:That Begins With OEW:That Ends With OSL:That Sounds Like	OOS:That Contains OSM:Contains In Order OEM:= OLE:<= OGE:>= OBW:That Begins With OEW:That Ends With OSL:That Sounds Like
`search_result_views`	When a search is performed in Organization Manager these are the possible display format(s) for the results. Any combination of these values is allowed. The absence of any one of these values disables that search result's view format.	table_view custom_view	table_view custom_view

Table 7 groupservcenterparams.xml

Parameter name	Description	Default Value	Possible Values
searchString MinimumLength	The integer value for the minimum number of characters that must be provided as the basis for a search. This overrides, for Organization Manager only, the value provided in the `oblixappparams.xml` file. NOTE: This parameter does not appear in the installed version of this file. If added by the user, the value here applies only to Group Manager.	3	Any positive, non-zero integer

Table 8 objservcenterparams.xml

Parameter Name	Description	Default Value	Possible Values
`navbar_bgcolor`	The background color for application navigation bar. The value is presented in the obbgcolor attribute of ObNavbar element.	#FFCC00	Any RGB value
`ObEnhanceSearchList`	Drop-down selection of Search Condition in Search toolbar. Name is a Search Condition understood by the application; the value is displayed on the selection menu.	OOS:That Contains OSM:Contains In Order OEM:= OLE<= OGE:>= OBW:That Begins With OEW:That Ends With OSL:That Sounds Like	OOS:That Contains OSM:Contains In Order OEM:= OLE:<= OGE:>= OBW:That Begins With OEW:That Ends With OSL:That Sounds Like
`search_result_views`	When a search is performed in Organization Manager these are the possible display format(s) for the results. Any combination of these values is allowed. The absence of any one of them disables that search results view format.	table_view custom_view	table_view custom_view
searchString MinimumLength	The integer value for the minimum number of characters that must be provided as the basis for a search. This overrides, for Organization Manager only, the value provided in the `oblixappparams.xml` file. NOTE: This parameter does not appear in the installed version of this file. If added by the user, the value here applies only to Organization Manager.	3	Any positive, non-zero integer

Table 9 gsc_wf_params.xml, osc_wf_params.xml, usc_wf_params.xml

Parameter name	Description	Default value	Possible values
`A compound list per workflow type`	This compound list contains detailed parameters for each of the workflow types shown in the **Possible Values** column. Under each workflow type there appears a set of `Actions compound lists`, as explained below.	None	CREATE_OBJECT DELETE_OBJECT CHANGE_ATTRIBUTE CERT_ENROLL CERT_RENEW CERT_REVOKE
`Actions compound list`	There is an action compound list for each permissible action for a workflow type. The compound list for workflow type will contain one action compound list per valid action for that workflow type. For example: CREATE_OBJECT will have compound lists for the following actions: initiate, self_registration, provide_info, approval, provide_approval, activate, commit, external_action, error_report. Under each of these there is a set of parameters and values, as described in the rest of this table.	None	initiate self_registration request cert_initiate_enroll cert_initiate_renew cert_initiate_revoke cert_generate cert_revoke provide_info change_info approval provide_approval change_approval activate deactivate commit error_report external_action
`archiveFileName`	File name of the archive file.	None	Correct file name
`deactivate archiveFileName`	File name of the deactivated users archive file.	None	Correct file name

Table 9 gsc_wf_params.xml, osc_wf_params.xml, usc_wf_params.xml

Parameter name	Description	Default value	Possible values
`exclude_attrs`	Exclude attribute(s) from showing up in relevant data	None	Attribute name in the schema. For SecureWay, gsc_wf _params.xml is replaced by gsc_wf _params-sw.xml during Setup. `obgroup` `puredynamic` is excluded in `CREAT_OBJECT`
`exit_condition`	A `ValNameList` which defines the two parameters: `false` `true`	None	`0` and `1`, respectively
`forcecommit`	Flag indicating whether the entry should be committed before the user action for this action, for example: activate, deactivate.	false	true false
`Notifee`	A ValList for which the member values may be any of the items in the **Possible Values** list. These are allowed roles for the person to be notified.	None	dns ob_self previous step owner current step participants next step participants initiator
`occurrence`	Allowed number of occurrences for each action.	None	1 n
`Participant`	A ValList for which the member values may be any of the items in the **Possible Values** list. These are allowed roles for the participant.	None	ob_any dns ob_self
`pre_action`	A ValList, which is a list of possible actions that may occur before this one.	None	any action name.
`relevant_data`	A ValList for which the member values may be any of the items in the **Possible Values** list. These are possible types of relevant data for this action.	None	required provisioned optional

Table 9 gsc_wf_params.xml, osc_wf_params.xml, usc_wf_params.xml

Parameter name	Description	Default value	Possible values
`subscription_policies`	A ValList for which the member values may be any of the items in the **Possible Values** list. These are a set of allowed subscription policies.	None	Subscription PolicyOpen Subscription PolicyOpenFilter Subscription PolicyControlled Workflow Subscription PolicyClosed
`useraction`	Flag indicating if user action is required for that particular action. For example, provide_info, approval, activate will have useraction flag set to true. Commit, external_action would have useraction as false.	None	false true
`wf_name`	A compound list of user-friendly names for the various workflow types.	None	Can be any user friendly string for the workflow type
`wfDateFormat`	Workflow date formats.	None	2 (mm/dd/yyyy) 3 (dd/mm/yyyy) 4 (dd/mmm/yyyy) 5 (mmm/dd/yyyy)

Table 9 gsc_wf_params.xml, osc_wf_params.xml, usc_wf_params.xml

Parameter name	Description	Default value	Possible values
wfDateSeparator	Single character used to separate the YMD parts of a date provided in `wfDateFormat`. If the parameter file does not specify a character for this parameter, then the default is used.	`/` (slash)	`/` (slash) `-` (hyphen) `.` (period) `,` (comma) (space)
initialStep	Signals if a step with that action can be the first step for that particular type of workflow. NOTE: Oblix does not recommend the user to change the values for these parameters. For example the users cannot make the commit step as the first step by simply setting its initialStep parameter to true. However, for a step that is permitted as a first step, the user can set its initialStep parameter to false. Basically, the users cannot add to the permitted first steps set, however they can remove items from it on a per workflow type basis.]	`false`	`true` `false`

Table 10 asynchparams.xml

Parameter Name	Description	Default Value	Possible Values
`asynch_user`	The DN of a user who is allowed to do asynchronous operation.	none	Any valid user DN
`mailer_sleep_time`	Duration of the sleep time for which the mailer goes to sleep, then wakes up to send the pending mail.	10	Any positive integer value, in seconds
`queuewaittime`	Queue wait time for the global mail queue.	10	Any positive integer value, in milliseconds

Table 11 querybuilderparams.xml

Parameter Name	Description	Default Value	Possible Values
`navbar_bgcolor`	This is used to set the back ground color of the Navigation Bar in Querybuilder.	#CC6666	Any RGB value
`ObQBOperators List`	Drop-down selection of Search Condition in QB filter toolbar. CND_EQ "Equals" CND_NEQ "Does not equal" CND_LTE "Greater than equal to" CND_GTE "Less than/equal to" CND_LT "Less than" CND_GT "Greater than" CND_CON "Contains" CND_DNC "Does not Contain" CND_PRE "Present" CND_NPR "Not Present" CND_BW "Begins With" CND_EW "Ends With" CND_DBW "Does not begin with" CND_DEW "Does not end with" CND_SLK "Sounds Like" CND_DSLK "Does not sound like"	As listed under Description	As listed under Description

Table 12 selectorparams.xml

Parameter Name	Description	Default Value	Possible Values
`navbar_bgcolor`	This is used to set the back ground color of the navigation bar in selector.	"#CC6666"	Any RGB value
`ObEnhance SearchList`	Drop-down selection of Search Condition in Search toolbar. Value is the Search Condition display name on the selection menu that is used by the application.	OOS: That Contains OSM: Contains In Order OEM: Equal to OLE: Less than or equal to(<=) OGE: Greater than or equal to(>=) OBW: That Begins With OEW: That Ends With OSL: That Sounds Like	The same, plus: ONE: Not equal to (!=)

Table 13 frontpageadminparams.xml

Parameter Name	Description	Default Value	Possible Values
`min_location _area`	The area allocated for the location GIF. This depends on each customer's location image.	400	A positive integer
`top_frame`	Name of the top frame in User Manager application.	_top	A frame name
`top_main_frame`	Name of the main frame in User Manager application	main_frame	A frame name

Table 14 globalparams.xml

Parameter Name	Description	Default Value	Possible Values
authUser Location	Position of the authuser variable within the request info. Netscape places the authuser variable in the variable section of the request info, while Site Minder places it in the request info headers.	headers	Auth user location in request info for example, `vars` (for Netscape) `headers` (for Siteminder)
backslash ReturnedAs	The escaped string representation of the '\' character as returned by the DS. This is used in context of the ObDPostalAddress display type. Since '$' is the delimiter in a postal address string, some directory servers return it in escaped format. In order to distinguish between a \ in an escaped string versus an actual \ in the value, the \ in the value is returned in an escaped format. For example, NDS returns it as "\\", Netscape returns it as "\5c". NOTE that when a '\' is part of the attribute value itself, it should be escaped and sent as "\5c" as per RFC 2252.	\5c	\5c or \\ and so on
browserNoCache	This parameter can be set to true or false. If it is set to true, (the default), then the browser will be instructed not to cache the page, if it is set to false, it will cache. You can set this value in the globalparams.xml file, or you can pass it on the URL.	`true`	`true` `false`
BypassAccess ControlForDir Admin	Indicates whether the attribute access control should be bypassed for directory administrators.	`true`	`true` `false`
cookieDomain	Domain used when setting any cookies. Default is the machine name. Usually used when the customer has set up something like DNS Round-Robin for better performance or server fail-over.	""	"" or for example oblix.com
cookieSeperator	Cookie delimiter used for compacting the various cookies. DO NOT CHANGE THE # VALUE.	# DO NOT CHANGE	# ONLY. DO NOT CHANGE.
cookieSizeLimit	Maximum cookie size.	4096	Integer value = 4096

Table 14 globalparams.xml

Parameter Name	Description	Default Value	Possible Values
disable_native _deactivate	If Directory is AD, NDS, or iPlanet5, when a user is deactivated, the application will utilize a directory-native deactivate feature to disable the account. This feature is enabled by default. The flag is to disable this feature.	true	true or false
dollar ReturnedAs	The escaped string representation of the '$' character as returned by the directory server. This is used in context of the ObDPostalAddress display type. Since '$' is the delimiter in a postal address string, some directory servers return it in escaped format. For example, NDS returns it as "\$", Netscape returns it as "\24". NOTE that when a '$' is part of the attribute value itself, it should be escaped and sent as "\24" as per RFC 2252.	\24	\24 or \$ etc.
exclusiveAutnCheckout	If a directory server does not support concurrent binds on the same LDAP connection, this parameter ensures that the binds are serialized on the connection. This ensures that multiple connections can be established and that the load is balanced on these connections. This value is set to true for NDS and cannot be changed. NDS does not support concurrent binds on a single LDAP connection. For any other directory that does not support concurrent binds on a single LDAP connection, you must add this parameter with a value of true to the globalparams file.	true	true false
ExcludeOCs ForTreeInApplet	When there are many users under the same parent node, the performance of the user interface control (a Java applet) that allows you to graphically expand the node is adversely affected. This parameter allows you to specify a list of objectclasses for which expansion should not be performed.	inetOrgPerson	Objectclasses that the customer wants to exclude
formZero Threshold	This parameter controls the space the NetPoint system allocates to a buffer.	1000	Integer value

Table 14 globalparams.xml

Parameter Name	Description	Default Value	Possible Values
HTML_Message _End_Tag	HTML support for Message Catalog changes. HTML_Message_End_Tag is the configurable end tag.	<StopHTML>	Any valid HTML tag
HTML_Message _Start_Tag	HTML support for Message Catalog changes. HTML_Message_Start_Tag is the configurable start tag.	<StartHTML>	Any valid HTML tag
IsADSIEnabled	If using ADSI instead of LDAP to connect to Active Directory, this parameter is set to TRUE.	None	true false
ListOf SupportedDS	This parameter lists all the supported data stores. NCSP4 is Netscape4.x LDAP server, Novell is NDS, and MSAD is Microsoft Active Directory.	NCSP4 Novell MSAD	The same as listed
locale_params	This parameter contains all the necessary input information for running NetPoint in different locale mode. charset is character set, language is current language, doUtfConversion indicates whether to do UTF conversion or not.	charset - iso-8859-1 language - En_US doUtfConversion - NO useLanguageSort - NO sortRulesFile - ""	charset—Any valid character set language—Any valid language doUtfConversion—NO or YES useLanguageSort—NO or YES sortRulesFile—"" or valid rules for sort file
logRequestUrl	If logRequestUrl is true, a URL is set to log request. It is used by WebPass.	false	true or false
maxDBAgent CacheSize	Define the DB agent cache size.	2000	Any positive integer
maxForRangedMember Retrieval	This parameter must be set to retrieve members from groups with a large number of static members. This parameter is used for Active Directory 2000 and Active Directory 2003.	1000	The default value is 1000, which is appropriate for Active Directory 2000. Tor Active Directory 2003, set this value to 1500.
nsAuthUser	Name of the authentication user variable while using Netscape Webserver or the IIS Web server	HTTP_OBLIX_ UID	Authentication user variable name. For example, auth-user (for Netscape) SM_USER (for Siteminder)

Table 14 globalparams.xml

Parameter Name	Description	Default Value	Possible Values
oisClientTimeout Threshold	How long (in seconds) one COREid Server attempts to contact another COREid Server before it considers it unreachable, in which case an error is logged.	60	An integer, representing number of seconds.
OutputFormat	Request Info output format, for use with PresentationXML.	default	`default`—Combine the XML and stylesheet at the server (server side processing). You can override this by including the `format` parameter in the Presentation XML request. `xml`—Send the XML and the stylesheet to the browser (client side processing). You can not override this in the PresentationXML request.
ResourceFilter SearchScope	The level of scope of search on a given searchbase.	1	1 indicates 1 level down, 2 as many levels as exist. Entry of any other value uses the default value (1).
sendMail Notification Enabled	Enables or disables notification events from workflows, attribute changes, and container limit events. The flag has no effect on bug or feedback emails since these are routed though the user's email client.	false	true or false
TimeToWaitFor ServiceThreads	A thread that wants to flush the osd and/or config db cache needs to wait for all other service threads to complete before flushing. This value is the maximum time the flush thread should wait - in seconds - before flushing. If all service threads complete before this time, then the flush thread will stop waiting and start flushing.	60	Integer value - greater than or equal to zero. (zero is legal but not a good idea; setting this value too low could lead to SEGV crashes)

Table 14 globalparams.xml

Parameter Name	Description	Default Value	Possible Values
UidInfoCache	UidInfoCache contains information about uid caching. UidInfoCache.maxNumElems - indicates the max number of uid to be cached. UidInfoCache.timeout - sets the time out of the uid cache. UidInfoCache.disabled - indicates whether to disable or enable the Uid info cache.	UidInfoCache .maxNumElems - 50000 UidInfoCache .timeout - 0 UidInfoCache .disabled - false	UidInfoCache .maxNumElems - Integer number UidInfoCache .timeout - time in seconds UidInfoCache .disabled - false or true
UseLDAPFor Authentication	In a pure ADSI environment, if this flag is enabled, Netpoint will use LDAP for authenticate calls. All other operations would go through ADSI.	None	`true` `false`
whichAttrIs Login	This parameter indicates which directory attribute is used to log into the NetPoint system.	HTTP_OBLIX_ LOGIN_VAR	Any directory attribute or HTTP_OBLIX_ LOGIN_VAR
whichVarIsOblixLang	This parameter specifies the name of the header variable that specifies the language of the request.	HTTP_OBLIX_ LANGUAGE	Header variable name
whichVarIsAcceptLang	This parameter specifies the name of the header variable in the user's browser that specifies the language of the request.	Accept-Langua ge	Header variable name
whichVarIsUser Type	Name of the HTTP header variable containing user type information. The value must to obnavigation.xml. This is additional support of navigation if usertype parameter is not in the URL, mainly for single sign-on.	HTTP_OBLIX_ USER_TYPE HTTP	Header variable name

Table 14 globalparams.xml

Parameter Name	Description	Default Value	Possible Values
`XMLStructure Cache`	This cache stores in memory the static portion of each XML document. XMLStructureCache.maxNumElems - Maximum number of elements to be stored in the cache (integer). XMLStructureCache.timeout - Number of seconds that an element remains in the cache. If this value is 0, then elements are never timed out of the cache. XMLStructureCache.disabled - If this argument is set to "true", the cache is disabled.	XMLStructure Cache.max NumElems - 20 XMLStructureCache. timeout - 0 XMLStructureCache.disabled - false	XMLStructure Cache.max NumElems - Any integer XMLStructure Cache.timeout - Any valid seconds XMLStructure Cache.disabled - false or true
`PortalIdCache`	PortalIdCache defines information that controls portalId caching. PortalIdCache.maxNumElems—Indicates the max number of portal Id's to be cached. PortalIdCache.timeout—Sets the time out of the portalId cache refresh. PortalIdCache.disabled—Indicates whether to disable or enable the PortalId cache.	`PortalIdCache. maxNumElems` - 250 `PortalIdCache. timeout` - 0 `PortalIdCache. disabled` - false	`PortalId Cache. maxNumElems` - Integer number `PortalId Cache. timeout` - time in seconds `PortalId Cache. disabled` - false or true
`ActiveDirectory`	True if the NetPoint Administrator selects Active Directory as the Directory Server type during COREid Server configuration, false otherwise.	None	`true` `false`

Table 14 globalparams.xml

Parameter Name	Description	Default Value	Possible Values
`XSLStylesheet CacheSize`	This controls the maximum number of stylesheets that will be held in the cache. A cached stylesheet is already in a binary form that can be used immediately for an XSL transformation to generate the requested page. If the stylesheet for the requested page is not in the cache, it must first be loaded from disk and processed by the XML parser before it can be used for a transformation. This step is time-consuming; caching the most frequently used pages can eliminate it and reduce the perceived latency. The trade-off is that cached binary stylesheets can be quite large. (Exactly how large depends on your stylesheet design.) An efficient strategy to conserve memory is to set this parameter slightly higher than the number of pages that you consider frequently used; all those stylesheets will be cached, and relatively infrequent ones can be brought into cache without flushing the common ones.	20	Any integer greater than zero. *Do not* use a value less than or equal to zero. If you do, an internal test value is used; this value is *not* zero.
`XSLStylesheet LiveUpdate`	This causes the following behavior when the stylesheet for the requested page is already in the stylesheet cache: true—Check timestamp on the top-level stylesheet file; if the file is newer, refresh the cache entry. false—Do not check; if the stylesheet is cached, use it. For developers and customer's modifying stylesheets, true is convenient, because you do not have to restart the server or artificially fill the cache in order to see the result of a stylesheet update. In a stable system, setting this to false eliminates an unnecessary file system access for cached stylesheets, and can result in better performance.	false	true or false

Table 15 oblixadminparams.xml

Parameter Name	Description	Default Value	Possible Values
`csv_field_delim`	CSV Field delimiter is used to separate two fields when generating reports.	`,` comma is used by default	
`csv_value_delim`	CSV value delimiter is used as to separate two values when generating reports.	`,` comma is used by default	
`config_meta _attr_applet_bg`	An RGB hexadecimal number that defines the configuration attributes background color.	cccccc	An RGB hexadecimal number defining color
`config_meta _attr_applet_fg`	An RGB hexadecimal number that defines the configuration attributes foreground color.	000000	An RGB hexadecimal number defining a color
`mime_type_file _location`	The location of the MIME type file.	../../admin/bin/ mime_types.lst	Should not be configured
`oblixNode`	The RDN of the node under which all the Oblix configuration information is stored. This is prefixed to the config DN that the customer gives during setup and the entire DN is the container for all Oblix data. For example, if the configuration DN was specified as "o=company,c=us", and the oblixNode parameter is given the value "o=configdata", then the oblix container DN is "o=configdata, o=company,c=us".	Default is none, that is. The parameter is not specified in file at all in the installed version of this file. Until specified otherwise during setup, the value is taken to be `o=oblix`.	Any valid RDN values such that they satisfy the container requirements of the parent node [the config DN].

Table 16 oblixappparams.xml

Parameter Name	Description	Default Value	Possible Values
`checkChangeAttributeEvenAllowModify`	For performance reasons, if a user has modify access to an attribute, applications do not check that the user is a participant in the *Change Attribute* workflow for that attribute. The rationale is: *since the user can modify the attribute, why bother taking the time to check this?* If this flag is *true*, and the user has modify access, applications *will* check that the user is a participant. So even if user can modify the attribute directly, Request button(s) will show up.	false	true or false
`csv_field_delim`	CSV Field delimiter is used to separate two fields when generating reports.	`,` comma is used by default	
`csv_value_delim`	CSV Value delimiter is used as to separate two values when generating reports.	`,` comma is used by default	
`enable_oaview`	Enable optional authentication view.	false	true for true false or any other string
`group_cgi`	The URL to get to a group application.	../../groupservcenter/bin/groupservcenter.cgi	../../groupservcenter/bin/groupservcenter.cgi
`group_view_program`	The program used to view a group profile (this is used to append to the URL as &program=view) or whatever program you want the application to go to (during cross application linking) view a group.	view	view (go to the Group Manager application for other options such as viewing member details, and so on)
`initial_search_advance`	Use the initial search as the first view when user wants to perform a search.	false	true for true false or any other string
`initial_search_advance_noof fields`	The number of fields to display for an initial advanced search. Use in conjunction with initial_search_advance.	3	Any positive integer

Table 16 oblixappparams.xml

Parameter Name	Description	Default Value	Possible Values
object_cgi	The URL to get to the Organization Manager application.	../../objservcenter/bin/objservcenter.cgi	../../objservcenter/bin/objservcenter.cgi
object_view_program	The program used to view an object profile (this is used to append to the URL as &program=view) or whatever program you want the application to go to (during cross application linking) view a object.	view	view (go to the group manager application for other options)
search_result_show_count	Show the count for the number of search results returned in a search operation.	false	true for true false or any other string
search_result_views	When a search is performed, these are the possible display format(s) for the results. Any combination of these values is allowed. Also the order of the search results side tabs depends on the order of the values listed. The absence of any one of these values disables that search results view format.	table_view custom_view	table_view custom_view
searchSame AttrAsOr	If the same attribute has provided multiple values in a search request, assume that it is an AND if set to false or an OR if set to true.	false	true for true false or any other string
searchString MinimumLength	The minimum number of characters that the end user needs to provide in order to perform a search operation. The value provided here can be overridden, for each of the Manager applications, by adding this parameter to the parameter file specific to the Manager application.	3	Any positive integer.
user_cgi	The URL to get to a user application such as User Manager.	../../userservcenter/bin/userservcenter.cgi	../../userservcenter/bin/userservcenter.cgi
user_view_program	The program used to view a user profile (this is used to append to the URL as &program=view) or whatever program you want the application to go to view a user (during cross-application linking).	view	view

Table 16 oblixappparams.xml

Parameter Name	Description	Default Value	Possible Values
validateAllDnViewMode	This is used to turn on/off the DN validation for the view mode for the values of all DN-type attributes. This is a Boolean param. If it is true, all DN attributes will be validated before being displayed to the user. By validation, what is meant is that the login user will only see those values of the DN that he has view access [on the class attribute of the objectclass of the DN] as well as localized access [that is, this DN falls under the user's searchbases with respect to the type of objectclass of the DN].	false	true/false
validateAllDn ModifyMode	This is used to turn on/off the DN validation for the view mode for the values of all DN-type attributes. This is a Boolean param. If it is true, all DN attributes will be validated before being displayed to the user in the form. Validation means that the login user will only see those values of the DN that he has view access [on the class attribute of the objectclass of the DN] as well as localized access [that is, this DN falls under the user's searchbases with respect to the type of objectclass of the DN]. Thus, the user will be allowed to add/remove only thost DNs that he has access to.	false	true/false

Table 16 oblixappparams.xml

Parameter Name	Description	Default Value	Possible Values
validateDnAttrs ViewMode	This is used to turn on or off the DN validation for the view mode for the values of the specified DN type attribute. This is a ValList param. You provide the list of attributes as a vallist. This parameter is used only if the param called "validateAllDnViewMode" is set to false. This allows attribute level validation, whereas the parameter "validateAllDnViewMode" provides global validation. DN attributes specified in this vallist are validated before being displayed. Validation means that the login user will only see those values of the DN that he has view access [on the class attribute of the objectclass of the DN] as well as localized access [that is, this DN falls under the user's searchbases with respect to the type of objectclass of the DN].	none	A vallist of DN type attributes. [Use ldap names and not display names]
validateDnAttrs ModifyMode	This is used to turn on or off the DN validation for the modify mode for the values of the specified DN type attribute. This is a ValList param. You provide the list of attributes as a vallist. This parameter is used only if the param called "validateAllDnModifyMode" is set to false. This allows attribute level validation, whereas the parameter "validateAllDnModifyMode" provides global validation. DN attributes specified in this vallist are validated before being displayed in the form. Validation means that the login user will only see those values of the DN that he has view access [on the class attribute of the objectclass of the DN] as well as localized access [that is, this DN falls under the user's searchbases with respect to the type of objectclass of the DN]. Thus the user will be allowed to add/delete only those DN values to which he has access to.	none	A vallist of DN type attributes. [Use ldap names and not display names]

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
groupservcenter _admin _application_info with sub parameters: VERSION CODE ID PROGRAM DESCRIPTION NAVBAR_GIF NAVBAR_GIF2 NAVBAR_GIFDIR	Specific information regarding the Group Manager Admin application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00 CODE=GMAD ID=groupservcenter _admin PROGRAM=../../ admin/bin/ front_page_admin.cgi?targetApplication = groupservcenter _admin DESCRIPTION= Group Manager Admin NAVBAR_GIF= OTABgroupmanager NAVBAR_GIF2= OTABgroupmanager 2 NAVBAR_GIFDIR=../../common/ui/style0	Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR _GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR _GIFDIR)
userservcenter _admin _application_info with sub parameters: VERSION CODE ID PROGRAM DESCRIPTION NAVBAR_GIF NAVBAR_GIF2 NAVBAR_GIFDIR	Specific information regarding User Manager Admin application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00 CODE=UMAD ID=userservcenter _admin PROGRAM=../../ admin/bin/ front_page_admin.cgi?targetApplication =userservcenter_admin DESCRIPTION= User Manager Admin NAVBAR_GIF= OTABusermanager NAVBAR_GIF2= OTABusermanager2 NAVBAR_GIFDIR= ../../common/ui/ style0	Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR _GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR _GIFDIR)

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
access_control _applet with sub parameters: applet _dimension _width applet _dimension _height column_width	This list contains customization values for dimensions of the Attribute Access Control applet.	applet_dimension _width =630 applet_dimension _height=765 column_width=135	A positive integer
access_front_page _admin _application_info with sub parameters: VERSION CODE ID PROGRAM DESCRIPTION NAVBAR_GIF NAVBAR_GIF2 NAVBAR_GIFDIR	Specific information regarding Access Administration application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00 CODE=AD30 ID=access_front_page_admin PROGRAM=../../../../../access/oblix/apps/admin/bin/front_page_admin.cgi? DESCRIPTION=Access Administration NAVBAR_GIF=T1TABaccessadmin NAVBAR_GIF2=T1TABaccessadmin NAVBAR_GIFDIR=../../common/ui/style0	Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR _GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR _GIFDIR)
applet _customizations	To help the system administrator to tweak dimensions for various applets used in the COREid System This compound List contains the following valname lists. workflow_definition_applet setsearchbase_applet delegate_admin_applet access_control_applet	According to the list	

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
`Apply _LostPwdMgmt`	Specify whether to apply lost password management.	Default parameter in params file is Yes. If no value is specified in the parameter catalog, then product assumes No.	Yes (case insensitive) all other values mean no
`certAttrs`	Attribute values that can show up on a certificate.	issuerDN validFrom validTill	(multi-valued parameter) issuerDN validFrom validTill SubjectDN PubKeyAlgID Version
checkuseris deactivated	When a user initiates an action, NetPoint can be set to check to see if that user is deactivated. By default, this check is disabled in order to reduce the number of reads of the directory. The check can be enabled by adding this parameter, and setting its value to `true`.	false	true false

`containment limit_applet with sub parameters:`
 	
`applet _dimension _width`
 	
`applet _dimension _height`
 	
`column_width`
	This list contains customization values for dimensions of the Containment Limit applet.	applet_dimension _width =805
		
		applet_dimension _height=467
		
		column_width=135
			A positive integer

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
`cookieBustLimit`	Number of people (such as in selector) that can be selected before the cookie size limit is exceeded. This depends greatly on the size of the DN for each entry, and upon the operating system. Suggested values are 15 or less for Active Directory, 25 or less for others.	30	A positive integer. If there are any Latin-1 characters in the user DN, then each such Latin-1 character should be counted as 3 characters (this is because Latin-1 characters are escaped to their %xx hex equivalent in the cookie)
`dateSep`	A character used to separate fields in a date value.	/	A single character
`dateType`	Different formats to display a date value.	ObMDYDate	ObMDYDate (12/31/1985), ObDMYDate (31/12/1985), ObDMonthYDate (31-Dec-1985), ObMonthDYDate (Dec-31-1985), ObIntegerDate (yyyy-mm-ddTh:mm:ss), ObISO8061Date (yyyy-mm-ddTh:mm:ssTZD or yyyymmddThhmmssTZD), where TZD = {+-}hh:mm)

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values

default_display_vals with sub parameters:
 	
`defaultDisplay Name`
`defaultDisplay Val`
	Display name for a no-operation, single-selection menu item and its corresponding value. This is used while creating a report.	defaultDisplay Name=None
		
		defaultDisplay Val=
			Any string

`defaultDisplay ResultVal`	Default number of values to be displayed in the display result for a search. It is used when the user first does a search, or if the user's cookie file is not available. Subsequent searches get this value from the user's cookie. This value also controls what is shown on Generate Reports, Incoming Requests, Outgoing Requests, and Monitor Requests pages in the COREid Server.	8	A positive integer

`delegate _admin_applet with sub parameters:`
 	
`applet _dimension _width`
 	
`applet _dimension _height`
 	
`column_width`
	This list contains customization values for dimensions of the Delegate Admin applet.	applet_dimension _width =630
		
		applet_dimension _height=665
		
		column_width=135
			A positive integer

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
front_page_admin_application_info with sub parameters: VERSION CODE ID PROGRAM DESCRIPTION NAVBAR_GIF NAVBAR_GIF2 NAVBAR_GIFDIR	Specific information regarding Identity Administration application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00 CODE=FPAD ID=front_page_admin PROGRAM=../../admin/bin/front_page_admin.cgi DESCRIPTION=IdentityAdministration NAVBAR_GIF=T1TABidentityadmin NAVBAR_GIF2=T1TABidentityadmin NAVBAR_GIFDIR=../../common/ui/style0	Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR_GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR_GIFDIR)
groupservcenter_application_info with sub parameters: VERSION CODE ID PROGRAM DESCRIPTION NAVBAR_GIF NAVBAR_GIF2 NAVBAR_GIFDIR WORKFLOW_ALLOWED	Specific information regarding Group Manager application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00 CODE=GM50 ID=groupservcenter PROGRAM=../../groupservcenter/bin/groupservcenter.cgi DESCRIPTION=Group Manager NAVBAR_GIF=T1TABgroupmanager NAVBAR_GIF2=T1TABgroupmanager NAVBAR_GIFDIR=../../common/ui/style0 WORKFLOW_ALLOWED=true	Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR_GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR_GIFDIR), WORKFLOW_ALLOWED (true means allowed, any other values mean not allowed)

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
`installed_apps`	Name of the applications that are enabled.	N/A	For the COREid System, the apps are: userservcenter (User manager), groupservcenter (Group Manager), objservcenter (Organization Manager)
`loginslack`	NetPoint expects the machine times for all Web Servers running Access Manager and COREid Server to be synchronized. If they are not, logging in to the Access Manager or the Access System Console is not possible. This parameter specifies a slack time in seconds by which the machine times may differ.	60	A positive integer (in seconds)
`max_url_length`	The maximum URL length for the specified browsers. The length is expressed in bytes.	netscape=4096 ie=1024	Netscape (A positive integer) IE (A positive integer)

objservcenter_admin_application_info with sub parameters:
 	
`VERSION`
`CODE`
`ID`
`PROGRAM`
`DESCRIPTION`
`NAVBAR_GIF`
NAVBAR_GIF2
`NAVBAR_GIFDIR`
	Specific information regarding Organization Manager Admin application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00
		CODE=OMAD
		ID=objservcenter_admin
		PROGRAM=../../admin/bin/front_page_admin.cgi?targetApplication=objservcenter_admin
		DESCRIPTION=Org. Manager Admin
		NAVBAR_GIF=OTABgroupmanager
		NAVBAR_GIF2=OTABgroupmanager2
		NAVBAR_GIFDIR=../../common/ui/style0
			Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR_GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR_GIFDIR)

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
objservcenter _application_info with sub parameters: VERSION CODE ID PROGRAM DESCRIPTION NAVBAR_GIF NAVBAR_GIF2 NAVBAR_GIFDIR WORKFLOW _ALLOWED	Specific information regarding Organization Manager application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00 CODE=OM50 ID=objservcenter PROGRAM=../../ objservcenter/bin/ objservcenter.cgi DESCRIPTION=Org. Manager NAVBAR_GIF =T1TABorgmanager NAVBAR_GIF2 =T1TABorgmanager NAVBAR_GIFDIR =../../common/ui/ style0 WORKFLOW _ALLOWED=true	Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR _GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR _GIFDIR), WORKFLOW _ALLOWED(true means allowed, any other values mean not allowed)
policyservcenter _application_info with sub parameters: VERSION CODE ID PROGRAM DESCRIPTION NAVBAR_GIF NAVBAR_GIF2 NAVBAR_GIFDIR	Specific information regarding Access Manager application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=1.0 CODE=PS10 ID=policyservcenter PROGRAM=../../../../ ../access/oblix/apps/ front_page/bin/ front_page.cgi DESCRIPTION= Access Manager NAVBAR_GIF= T1TABaccessmanag er NAVBAR_GIF2= T1TABaccessmanag er NAVBAR_GIFDIR= none	Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR_GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR _GIFDIR)

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
setsearchbase _applet, with sub parameters: applet _dimension _width applet _dimension _height column_width	This list contains customization values for dimensions of the Set Searchbase applet.	applet_dimension _width =650 applet_dimension _height=740 column_width=135	A positive integer
`showReplication Warnings`	This parameter is used to decide whether to display replication-related warnings (such as "Your changes may not be immediately available.") after any of the following operations: modify or add attributes, create ticket, process ticket, change style, modify or add location.	true	true or false

sysmgmt_application _info
with sub parameters:
 	
`VERSION`
`CODE`
`ID`
`PROGRAM`
`DESCRIPTION`
`NAVBAR_GIF`
NAVBAR_GIF2
`NAVBAR_GIFDIR`
	Specific information regarding System Admin application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00
		CODE=SMAD
		ID=sysmgmt
		PROGRAM=../../ admin/bin/ front_page_admin.c gi?targetApplication =sysmgmt
		DESCRIPTION= System Admin
		NAVBAR_GIF= OTABsystemadmin
		NAVBAR_GIF2= OTABsystemadmin2
		NAVBAR_GIFDIR= ../../common/ui/ style0
			Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR _GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR _GIFDIR)

`system_consoles`	The application to appear on the System Console.	front_page_admin	(multi-valued parameter) front_page_adm in policyservcenter access_front _page_admin

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values
top_frame	Name of the top frame in the Front Page application.	_top	A frame name (eg._top)
top_main_frame	Name of the main frame in the Front Page application.	main_frame	A frame name (for example, main_frame)
userservcenter _application_info with sub parameters: VERSION CODE ID PROGRAM DESCRIPTION NAVBAR_GIF NAVBAR_GIF2 NAVBAR_GIFDIR WORKFLOW _ALLOWED	Specific information regarding User Manager application. The listed parameters define the version of the application running, the code used for license checking, relative path of the application, mouseover message for the application, name of the GIF used on the top navigation bar, and the relative path to the GIF used on the top navigation bar.	VERSION=5.00 CODE=UM50 ID=userservcenter PROGRAM=../../userservcenter/bin/userservcenter.cgi DESCRIPTION=User Manager NAVBAR_GIF=T1TABusermanager NAVBAR_GIF2=T1TABusermanager NAVBAR_GIFDIR=../../common/ui/style0 WORKFLOW_ALLOWED=true	Possible customization for Description (any text string), NAVBAR_GIF (any gif name with .gif that exists in the NAVBAR_GIFDIR), NAVBAR_GIF2 (any gif name with .gif that exists in the NAVBAR_GIFDIR), WORKFLOW_ALLOWED(true means allowed, any other values mean not allowed)
ssologouturl	This parameter overrides the SSO Logout URL parameter configured in the Access System Console	None	Any valid URL that does the single sign-on logout.

Table 17 oblixbaseparams.xml

Parameter Name	Description	Default Value	Possible Values

`workflow _definition _applet, with sub parameters:`
 	
`applet _dimension _width`
 	
`applet _dimension _height`
 	
`column_width _workflowdef`
 	
`column_width _workflow _targetdef`
 	
`column_width _workflow _stepdef`
 	
`column_width _participant _notifee`
	This List contains customization values for dimensions of the workflow applet. This includes the three screens in the workflow creation, which are workflow definition, target definition and step definition. The column_width parameters apply to the left column of all the respective applets.	applet_dimension _width = 650
		
		applet_dimension _height=625
		
		column_width _workflowdef=160
		
		column_width _workflow _targetdef=160
		
		column_width _workflow _stepdef=160
		
		column_width _participant _notifee=100
			A positive integer

Table 18 appdbparams.xml

Parameter Name	Description	Default Value	Possible Values
`osdcache: warmupcache`	Warms up the OSD cache.	true	no/false (do not warm up) anything else (warm up)

Table 19 configdbparams.xml

Parameter Name	Description	Default Value	Possible Values
`enableLDAP Referral`	When LDAP server returns a referral, controls whether the referral is automatically chased.	true (automatically chase referral)	true (automatically chase referral) false (do not chase referral)

Table 20 userdbparams.xml

Parameter Name	Description	Default Value	Possible Values
`allow_non_rdn _modifications`	If this parameter is set to true, then modifying an attribute that is part of the DN will effect the DN itself and will result in moving the directory entry to a different subtree. This only applies to attributes that make up the non-RDN portion of the DN. For example, ou, o, and c in the DN "cn=John Smith, ou=Corporate, o=Company, c=US".	false (do not move the entry)	true (allow moving) false (do not move the entry)
`default_policy`	Default policy for access control to person object when no policy is found.	false (Deny Access)	true (Allow Access) false (Deny Access)

Table 21 groupdbparams.xml

Parameter name	Description	Default Value	Possible Values
`allow_non_rdn_modifications`	If this parameter is set to true, then the user can modify an attribute that is part of the DN, if they have modification rights. This check is imposed because non-RDN modification will affect the DN itself and will result in moving the directory entry to a different subtree. This affects referential integrity issues; this parameter allows the administrator to prevent such operations. This only applies to attributes that make up the non-RDN portion of the DN. For example, ou, o, and c in the DN "cn=John Smith, ou=Corporate, o=Company,c=US".	false (do not allow non-RDN modifications)	true (allow non-RDN modifications) false (do not allow non-RDN modifications)
`default_policy`	Default policy for access control to generic or location objects when no policy is found.	false (Deny Access)	true (Allow Access) false (Deny Access)
`default_subscription_policies`	Selects which of the four subscription policies supported by Group Manager are available. The policies are displayed at the time of workflow definition for create group. In the workflow definition, the user can select the subscription policies he wants to allow for groups that are created using this workflow definition. Then at the time of the actual create operation by the end-user, these options are shown in the Subscription Policy field, as a drop-down list, from which the end-user is supposed to select one policy that he wants to apply to this group. Note that the subset of the policies that are selected during workflow definition is also stored in each group entry created using that workflow, in an attribute hidden from the user. Later on, if the user wants to modify the subscription policies, then the values are obtained form this hidden attribute and again shown in the single-selection list.	All of the possible values are made available by default.	`SubscriptionPolicyOpen` (Automatic, no approval necessary) `SubscriptionPolicyOpenFilter` (Automatic if new member satisfies filter, no approval necessary `SubscriptionPolicyControlledWorkflow` (Needs approval through workflow) `SubscriptionPolicyClosed` (nobody can subscribe to this group)

Table 21 groupdbparams.xml

Parameter name	Description	Default Value	Possible Values
`default_subscription_policy`	Default policy for group subscription when no policy is found in the group entry.	`SubscriptionPolicyClosed`	The four allowed policies are: `SubscriptionPolicyOpen` (Automatic, no approval necessary) `SubscriptionPolicyOpenFilter` (Automatic if new member satisfies filter, no approval necessary) `SubscriptionPolicyControlledWorkflow` (Needs approval through workflow) `SubscriptionPolicyClosed` (nobody can subscribe to this group) Please refer to the parameter `default_subscription_policies` in this same table for more information.

Table 21 groupdbparams.xml

Parameter name	Description	Default Value	Possible Values
`extra_group_filter`	An LDAP filter. This filter, if so specified, is used by Group Manager to further qualify group searches. This filter may contain an Oblix rule substitution.	ou=ou. The meaning of this filter is that group searches will be further qualified so that the group must have the same ou value as the user. For example, if the user belongs to ou=corporate, the filter substitution will result in a filter of ou=corporate, which will be used to further qualify group searches.	Any valid LDAP filter, which may or may not contain a valid rule substitution NOTE: Any characters that are valid syntax for an LDAP filter, but are also xml markup, must be specified as entity references.
`max_filter_conditions`	This parameter can be used to control the length of the filter that gets used when doing group queries. It is an integer that says how many elements can make up the filter. The Group Manager application uses a search algorithm to minimize the number of searches done. It uses OR logic to combine multiple filters (essentially queries) into one large filter. But every directory server has its own limitations on the length of a filter used in doing the LDAP searches. This parameter allows the administrator to tune it according to the directory server used.	20	Any integer value, depending on what the directory server is able to handle
`use_extra_group_filter_expansion`	Indicates whether or not to use the extra_group_filter to further qualify group searches in group expansion.	false	true or false

Table 21 groupdbparams.xml

Parameter name	Description	Default Value	Possible Values
`use_extra_group_filter_mygroups`	Indicates whether or not to use the extra_group_filter to further qualify group searches in the MyGroups Profile.	false	true or false
`user_defined_unique_member`	This parameter is applicable to IBM SecureWay. In the SecureWay schema, a uniquemember attribute is required in the schema. Deactivating a user who is also the last member of a group causes an objectclass violation if the deactivation is done through User Manager. Therefore, User Manager attempts to replace this soon-to-be deactivated user with an entry for the Directory Administrators group. This parameter is used in place of the Directory Administrator group, if specified.	None	Any valid dn

Table 22 objectdbparams.xml

Parameter Name	Description	Default Value	Possible Values
`allow_non_rdn _modifications`	If this parameter is set to true then modifying an attribute that is part of the DN will effect the DN itself and will result in moving the directory entry to a different subtree. This only applies to attributes that make up the non-RDN portion of the DN. For example, ou, o, and c in the DN "cn=John Smith, ou=Corporate, o=Company, c=US". Unlike similar parameter in groupdbparams.xml and userdbparams.xml, this parameter is configured per objectclass.	false (do not move the entry) per objectclass	true (allow moving) per objectclass false (do not move the entry) per objectclass
`default _containment _policy`	Default policy for Containment Limit when no policy is found.	false (Do not Allow Create)	true (Allow Create) false (Deny Create)
`default_policy`	Default policy for access control to generic or location objects when no policy is found.	false (Deny Access)	true (Allow Access) false (Deny Access)

Table 23 workflowdbparams.xml

Parameter name	Description	Default value	Possible values
`qs_state_ groupservcenter`	Controls whether Quickstart is enabled for Group Manager.	`true`	`true` `false`
`qs_state_ objservcenter`	Controls whether Quickstart is enabled for Object Manager.	`true`	`true` `false`
`qs_state_ userservcenter`	Controls whether Quickstart is enabled for User Manager.	`true`	`true` `false`
`WfDefCache Disabled`	Determines if the workflow caches are to be disabled or not.	false	true false
`WfDefCache MaxNoOfElmts`	Maximum number of allowed elements in each of the workflow caches.	25	Unsigned integer
`WfDefCache Timeout`	Timeout for each individual element in the cache.	0	Long integer

Table 23 workflowdbparams.xml

Parameter name	Description	Default value	Possible values
`WfDefMaxNumStepDefFiltersPerSearch`	Determines the maximum number of step definition filters that can be used in each search. If the final number of filters is more than this specified value then multiple searches will be done.	None	Integer
`WfInstanceNotRequired`	A flag indicating if a single-user-action step workflow instance should be written to the directory server. This flag enables you to not save workflow instances if they are based on a single user action step and are not required later (for example, for auditing) and improve workflow runtime performance. False—Write workflow intances to the directory server. True—Do not write to the directory server, unless otherwise required by the workflow definition.	`false`	`true` `false`

Table 24 ldapappdbparams.xml

Parameter Name	Description	Default Value	Possible Values
`ListOfDS AttributesFor Filter Substitution`	List of directory server read-only system attributes utilized for ACL filter substitution. These attributes values do not return unless the directory server specifically queries for them. The list is entered as a ValList, in the form <ValList ListName="ListOfDSAttributesForFilter Substitution"> <ValListMember Value="entrydn" Operation="Add"/> </ValList>	nothing	List of attributes such as entrydn, creatorsname, password expirationtime
`osdcache: hashsize`	The hash size for the cache.	3001	Any positive integer (preferably a prime number)

Table 25 ldapconfigdbparams.xml

Parameter Name	Description	Default Value	Possible Values
`dynamic Auxiliary`	Set objectclass. This is only used for AD: AD does not allow the use of auxiliary class in the objectClass attribute.	false	true false
`groupspecial Attrs`	Used to cache in attributes for group class.	The cn attribute is derived from the auxiliary class mailrecipient, and hence does not show up on the list of required attributes. Also, sAMAccount Name attribute is cached by default.	Any valid attribute names.
`bind-dn password`	Bind DN, and password	none	Any valid string value for each

Table 25 ldapconfigdbparams.xml

Parameter Name	Description	Default Value	Possible Values
`specialAttrs`	Used to cache in attributes for person class.	sAMAccount Name attribute is cached.	Any valid attribute names
`useOIDNaming Attribute`	If the oidnamingattribute flag is set, convert the name to oid. Currently, this flag is only set in the case of Active Directory.	false	true false

Table 26 basedbparams.xml

Parameter Name	Description	Default Value	Possible Values
default_policy	Default policy for access control to any object. If the driving application database does not override this parameter, the default set here is assumed.	false (Deny Access)	true (Allow Access) false (Deny Access)
doAccessServer Flush	This signals that the AccessGate client has been configured on the OIS server and it can now begin to send user flush requests to the Access System, using the AM API.	false	true false
enableAllow AccessCache			
SelfReg Generates SSOCookie	This tells the Access System to automatically logon the requester right after self-registration if the person is activated. To do this, the settings for SR_SSOCookieMethod and SR_SSOCookieURL parameters must also be specified in this file.	false	true false
SR_SSOCookie Domain	This is one of the ObSSOCookie generation parameters. If no value is specified for this parameter, the ObSSOCookie is not associated with a particular domain.	None	An valid domain name, for example oblix.com
SR_SSOCookieIP	One of the ObSSOCookie generation parameters. If no value is specified for this parameter, the client IP will be used.	None	Any of the IP or IP addresses, if any, specified in the IPValidation Exceptions parameter in the Access System file WebGate Static.lst
SR_SSOCookie Method	Access SDK query parameter, used with self-registration. This parameter, along with the SR_SSOCookieURL parameter, is used by the Access SDK to determine the URL and method that are protected. The SSOCookie will not be generated if this value is not specified.	GET	Any one of the HTTP Request Methods that are protected by the Access System

Table 26 basedbparams.xml

Parameter Name	Description	Default Value	Possible Values
SR_SSOCookie Path	One of the ObSSOCookie generation parameters. This parameter will be used to generate ObSSOCookie. If none is specified, / will be used.	/	/ or any URL path
SR_SSOCookieURL	Access SDK query parameter, used with self-registration.This parameter, along with the SR_SSOCookieMethod parameter, is used by the Access SDK to determine the URL and method that are protected The SSOCookie will not be generated if this value is not specified.	/identity/ oblix	Any URL protected by the Access System

Table 27 ldapreferentialintegrityparams.xml

Parameter Name	Description	Default Value	Possible Values
`Objectclasses AndAttributesTo Do Referential Integrity`	This compoundlist contains a set of ValList elements named after object classes. Each ValList may be empty or may contain ValListMember elements named after attributes belonging to the object class. The object classes listed are those that NetPoint will update whenever an entry is renamed (such as its DN changed). The attributes listed for each object class are of type DN, and thus may refer to the entry which is being renamed. If no attributes are listed for a particular object class, NetPoint queries the schema to find *all* the DN attributes for that object class. If there *is* an attribute list, then only the listed attributes are used for the referential integrity check.	See the following table for a list of objectclasses and attributes.	Any valid objectclass with DN syntax attributes. NOTE: In order for NetPoint to work correctly, the default values should NOT be changed. You should only *add* your own objectclass and attributes to this list.
`references_to _non_existing _entries _allowed`	Determines how to deal with a reference to a non-existent entry. Since AD and Novell automatically remove references to non-existent entries, this parameter should be set to false for those Directory Servers. The Netscape/iPlanet DS does not; NetPoint adjusts the reference as you direct.	`false`	AD—Set to `false` Novell—Set to `false` Netscape/ iPlanet: • Set to `false` to have NetPoint update DN attributes that point to an entry being renamed • Set to `true` to have NetPoint *not* update DN attributes referring to an entry being renamed

Table 27 ldapreferentialintegrityparams.xml

Parameter Name	Description	Default Value	Possible Values
`referential_integrity_using`	Determines the responsibility for renaming a DN. The Active Directory and Novell directory servers do this automatically, `ds` is therefore the proper entry. Netscape does not, leaving it to NetPoint to make the change; this is indicated by the parameter value `oblix`. These values are set by the installation process and must not be changed by the user.	Varies with the Directory Server, defined at install time.	AD—Set to `ds` Novell—Set to `ds` Netscape—Set to `oblix`
`unique_value_attrs`	Specify a list of attributes whose values need to be unique under the configured directory server namespace. Necessary values vary with the brand of directory server. The **Possible Values** column shows the required entries; users may add additional attributes.	`uid`	Novell—Eemove list AD—Add one ValListMember, `sAMAccountName` Netscape—Leave the default ValListMember, `uid`

Here are the attributes referred to above, under `ObjectclassesAndAttributesToDoReferentialIntegrity:`

Table 28 ObjectClass Attributes for Referential Integrity

ObjectClass	Attributes
`groupofuniquenames`	uniqueMember owner seeAlso
`inetOrgPerson`	manager secretary
`oblixattribute access`	obmodifyaccessuid obviewaccessuid obnotifyuid
`oblixAuxLocation`	oblocationdn
`oblixcreatedeleteaccess`	obaccessuid obnotifyuid

Table 28 ObjectClass Attributes for Referential Integrity

ObjectClass	Attributes
oblixGenericResource AuxClass	obResourceUid
oblixgroup	obgroupadministrator obgroupcreator
oblixGroupResource AuxClass	obResourceUid
oblixlocation	obparentlocationdn
oblixorgperson	obindirectmanager oblocationdn
oblixPolicyCondition	obpolicyconditionUid obpolicyconditiongroup
oblixUserResourceAuxClass	obResourceUid

Table 29 webpass.xml

Parameter Name	Description	Default Value	Possible Values
`debug`	Indicates whether or not the WebPass client should be in debug mode and write debug information to the debug file.	false	true—Use debug mode false—Do not use debug mode
`failover Threshold`	The number of COREid Server connections that the WebPass client will attempt to keep alive. If the number of connections falls below the failoverThreshold, the WebPass client will attempt to open additional connections until the number of open connections equals the failoverThreshold. To meet the failoverThreshold, the WebPass client will use COREid Servers first from the primary server list, then from the secondary server list.	1	Any number
`id`	Unique identifier for WebPass client plug-in.	webpassdefault	Any
`maxConnections`	The maximum number of connections to COREid Servers.	1	Any number
`maxSessionTime`	The time an COREid Server connection will remain open in hours.	24	Any number

Table 29 webpass.xml

Parameter Name	Description	Default Value	Possible Values
`primary_server_list`	List of primary COREid Servers. Each list entry is a triplet of host, port, numConnections.	The triplet (for example, defaulthost, 6022, 1).	Any valid triplet of (host, port, numConnections): host—The host on which the primary COREid Server resides port—The port on the host on which the primary COREid Server listens numConnections—The number of connections that the WebPass client can open to a particular primary COREid Server.
`refresh`	Indicates whether or not the WebPass client configuration file, `webpass.xml`, should be periodically updated with the configuration information stored in the directory.	true	true—The update should occur false—No update should occur

Table 29 webpass.xml

Parameter Name	Description	Default Value	Possible Values
`secondary_server_list`	List of secondary COREid Servers. Each list entry is a triplet of (host,port,numConnections)	None	Any valid triplet of (host,port,numConnections) host—The host on which the secondary COREid Server resides port—The port on host on which the secondary COREid Server listens numConnections—The number of connections that the WebPass client will open to a particular secondary COREid Server

Table 29 webpass.xml

Parameter Name	Description	Default Value	Possible Values
`security`	The mode of transport security used for WebPass client and COREid Servers. `open` -transport security mode where no authentication and no encryption is performed. The WebPass client does not demand any proof of the COREid Server's identity, and the COREid Server accepts connections from all WebPass clients connected to it. `simple` - transport security mode where communication between the WebPass client and the COREid Server is encrypted using TLS v1 (Transport Layer Security, RFC 2246). Webpass and COREid Server authenticate one another using a global password, which must be the same across installations. `cert` - transport security mode under which the data transferred between points is encrypted using SSLand a public key certificate.	open	open simple cert as described in the **Description** column.
`sleepFor`	A time interval in seconds. After each interval, the WebPass client will update its configuration if the refresh flag is set to true. Also, the interval after which the WebPass client will do its failoverThreshold calculation and open additional connections, if necessary.	60	Any number.

Table 30 overridedbprofile.xml

Parameter Name	Description	Default Value	Possible Values
`list of agents`	List of agents for which the default values obtained from the directory server are to be overridden. Each list has a list name that should be the same as the agent for which the connection parameters are required to be overridden. Each agent should be accompanied by the following (host, port, secureport) This is used in the case where one directory server replicates another, and the user wants to use the replicant. An example of this file is installed at: *COREid_install_dir*`/identity/oblix/data/common` You must change the content of the file and move it to: *COREid_install_dir*`/identity/oblix/data.ldap/common` in order for it to take effect.	none	A valid agent name along with the following three parameters: `host`—Hostname for the directory server `port`—Port at which the directory server listens for open LDAP connections `secureport`—Secure port for the DS

Table 31 accessdb.xml, appdb.xml, configdb.xml, obgroupdb.db.xml, obobjectdb.xml,userdb.xml, webresrcdb.xml, workflowdb.xml, ticketdb.xml

Parameter Name	Description	Default Value	Possible Values
`ldapRootDN`	Bind dn.	Specified during the setup	Any valid dn
`ldapRootPasswd`	Bind password.	Specified during the setup	Any password
`ldapServerName`	LDAP host name for this database.	Specified during the setup	Any valid host name
`ldapServerPort`	LDAP port number.	Specified during the setup	Any valid port number
`ldapSizeLimit`	Client side size limit.	0	Any valid integer
`ldapTimeLimit`	Client side time limit.	0	Any valid integer

Table 31 accessdb.xml, appdb.xml, configdb.xml, obgroupdb.db.xml, obobjectdb.xml,userdb.xml, webresrcdb.xml, workflowdb.xml, ticketdb.xml

Parameter Name	Description	Default Value	Possible Values
`workflow DefinitionBase` (only in `workflowdb.xml`)	The base dn where workflow definitions are stored.	None (obcontainer= workflowDefinitions	Any valid dn
`workflow InstanceBase` (only in `workflowdb.xml`)	The base dn where workflow instances are stored.	None (obcontainer= workflowInstances under oblix tree)	Any valid dn
`xmlns`	Oblix xml name space.	http://www.oblix.com	`http://www.oblix.com`

Table 32 adsi_params.xml (Active Directory Services Interface Parameters)

Parameter Name	Description	Default Value	Possible Values
`sizeLimit`	Integer value that limits the number of query results returned for authentication.	0	Do not change this value.
`timeLimit`	Integer value that limits the number of seconds before a query times out.	0	Any positive integer
`pageSize`	Page size of results that ADSI request from the server.	100	Any positive integer
`useImplicitBind`	Which credentials to use.	0	0—Implicit Credentials 1—Explicit Credentials 2—Use User Principal Name
`adsiCredential`	An LDAP specification of a user, such as “cn=Administrator,cn=users,dc=myhost,dc=mydomain,dc=com”.	None	Valid credential
`adsiPassword`	An encoded text string representing the LDAP user’s password.	None	Valid password

Table 32 adsi_params.xml (Active Directory Services Interface Parameters)

Parameter Name	Description	Default Value	Possible Values
`useGCForAuthn`	Flag, asks the question: do you want to use the Global Catalog for authentication. If set to true, users may not be able to login until user accounts are replicated to the Global Catalog from the respective domain controllers.	false	true false
`useDNSPrefixedLDAPPaths`	To prefix the domain name to LDAP strings, a new parameter has been added to the adsi_params.xml and adsi_params.lst files. By default this parameter is not in adsi_params.xml. Before running setup, this parameter has to be manually added and set to true for the COREid server. You do not need to set service login credentials.	None	true false
`encryption`	When set to true, this flag encrypts the traffic between the Netpoint servers and Directory Server. When set to true, the SSL port (636) on Active Directory should be enabled. The rootCA certificates must have been installed in the local store for Trusted Certificate Authorities. This flag is applicable for authentications in all bind modes, and for all directory server traffic for explicit bind types (1 and 2). Note that password change on Active Directory always goes through the SSL port (636), irrespective of what the encryption flag is set to.	false	true false
`aynchronous Search`	Flag, asks the question: shall ADSI operate in its default mode, enabled to perform asynchronous searches? If set to false, it does synchronous searches.	true	true false

D Configuring COREid System Navigation

The NetPoint COREid System ships with an interface that supports four types of users: End User, Delegated Admin, Delegated Identity Admin, and NetPoint Admin. Each user type has different rights and is limited to different levels of NetPoint functionality. When users log in to NetPoint, they will be presented with a series of screens, a *navigation system*, that is defined for their user type.

This system can be modified to:

- Support new user types
- Select the screens to be shown, and determine the order in which they are presented
- Specify a default user type.

This Appendix describes `obnavigation.xml`, the configuration file that controls the navigation system, and explains how to work with it.

Overview

The COREid System uses the `obnavigation.xml` file as a guide to build the OutPutXML. PresentationXML uses OutPutXML to build the *NetPoint Navigation Bar*, the top two lines of each page. It includes the application name, help and logout buttons, and the various tabs to select other modules within the application. The stylesheet of course provides the final definition of how to display this information, but `obnavigation.xml` determines its content. The interaction with the stylesheet is described in more detail under “Designing the GUI with PresentationXML” on page 15.

Obnavigation.xml File

When you installed the COREid System you put it into a directory which we'll call the *COREid_install_dir*, for example:

/var/NetPoint/Identity/Oblix (UNIX)

or

C:/NetPoint/Identity/Oblix (Windows NT)

The obnavigation.xml file is installed under this, in the directory:

COREid_install_dir/identity/oblix/apps/common/bin

The file is provided in an XML format, the schema for which is provided under "File Schema" on page 260.

File Content

The following is a part of the installed `obnavigation.xml` file showing all the element types that are discussed in the table below:

```
<?xml version="1.0" ?>
<ObNavigation defaultUserType="systemAdmin">
<ObHierarchy name="oblix" elementName="ObNavbar"
  userType="endUser" obdisplayName="End User"
    bgcolor="CCCC66">
<ObCollection name="ObMisc">
    <ObLink appName="common" name="T1help" />
    <ObLink appName="common" name="T1about" />
    ObLink appName="common" name="T1logout" />
</ObCollection>
<ObCollection name="ObApps">
    <ObLink appName="common"
        name="userservcenter_application_info"
        elementName="ObApplication">
      <ObCollection name="ObTitle">
      <ObLink appName="userservcenter"
               name="T1TABusermanager" />
      </ObCollection>
      <ObCollection name="ObFunctions">
        <ObLink appName="userservcenter"
            name="MyProfile" />
        <ObLink appName="userservcenter"
            name="Report">
          <ObCollection name="ObReportFunctions">
            <ObLink appName="userservcenter"
```

```
                    name="generateReport" />
                  <ObLink appName="userservcenter"
                    name="viewPredefinedReports" />
                </ObCollection>
              </ObLink>
              <ObLink appName="userservcenter"
                   name="Workflow">
                <ObCollection
                       name="ObWorkflowFunctions">
                  <ObLink appName="userservcenter"
                    name="wfOutgoingRequest" />
                </ObCollection>
              </ObLink>
            </ObCollection>
    </ObLink>
    ...
    <ObLink appName="common"
           name="groupservcenter_application_info"
           elementName="ObApplication">
    ...
    <ObLink appName="common"
           name="objservcenter_application_info"
           elementName="ObApplication">
    ...
    <ObLink appName="common"
           name="corpdir_application_info"
           elementName="ObApplication">
    ...
</ObCollection>
....
</ObHierarchy>
....
</ObNavigation>
```

Elements in this file are the following:

Element Name	Description	Example

`ObNavigation`	This is the root element for the XML structure.
	
	It contains one attribute:
	
	`defaultUserType` - This specifies the default user type. The value entered for the attribute must match one of the user types defined in the rest of the file.
	
	The `ObNavigation` element contains one or more `ObHierarchy` elements.
		`<ObNavigation defaultUserType= "systemAdmin">`
		`...`
		`</ObNavigation>`

`ObHierarchy`	The `ObHierarchy` element defines the navigation structure, as a nested hierarchy, for a user type.
	
	It contains five attributes:
	
	• `name`—Reserved for future enhancements, currently not used.
	• `elementName`—The element name in the Output XML that contains the navigation information. The installed stylesheets expect its value to be `ObNavBar`; change this value only if you are willing to do extensive stylesheet changes.
	• `userType`—A unique value specifying the user type that uses this hierarchy.
	• `obdisplayName`—The display name for this user type.
	• `bgcolor`—Reserved for future enhancements, currently not used.
	
	Each `ObHierarchy` element contains one or more `ObCollection` elements.
		`<ObHierarchy name="oblix"`
		`elementName="ObNavbar"`
		`userType="endUser"`
		`obdisplayName="End User"`
		`bgcolor="CCCC66">`
		`...`
		`</ObHierarchy>`

Element Name	Description	Example

`ObCollection`	The `ObCollection` element is a grouping of links. The collection itself does not enforce any navigation structure. Instead, it is a conceptual element used to group links with common themes together.
	
	It contains one attribute:
	
	• `name`—This matches an element name in the Output XML. The installed stylesheets expect this to be either `ObMisc` or `ObApps`.
	
	Each `ObCollection` element contains one or `ObLink` elements.
		`<ObCollection name="ObMisc">`
		...
		`</ObCollection>`

`ObLink`	The `ObLink` element represents a link, where a link is a set of information that enables user navigation to a certain functionality within NetPoint.
	
	Some `ObLink` elements (but not all) are allowed to contain `ObCollection` elements. This means that, rather than directly providing functionality, the link presents the users with another set of links for navigation.
	
	Each ObLink contains the following attributes:
	
	• `appName` and `name`—These must be provided as a pair, meaning within the named application allow use of the named functionality. There is a limited set of valid combinations, predefined within the COREid System. See the table of “Valid ObLink Combinations” on page 264 for the full list. `name` values which are allowed to contain nested `ObCollection` elements are marked with an * in this list.
	• `elementName`—This element is optional. It provides a description for `ObLink` elements which contain nested `ObCollection` elements.
		`<ObLink appName="common"`
		`name=`
		`"userservcenter_`
		`application_info"`
		`elementName="ObApplication">`
		...
		`</ObLink>`

File Schema

Following is the schema describing the logical structure of the `obnavigation.xml` file. This schema definition is *not* provided as part of the COREid installation files. See the reference provided in “XML Background” on page 171 for more information on XML schema structures.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
   elementFormDefault="qualified">
<xsd:element name="ObCollection">
   <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref=
            "ObLink" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" use="required"
            type="xsd:string"/>
   </xsd:complexType>
</xsd:element>
<xsd:element name="ObHierarchy">
   <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref=
            "ObCollection" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string"
            use="required"/>
        <xsd:attribute name="elementName"
            type="xsd:string" use="required"/>
        <xsd:attribute name="userType"
            type="xsd:string" use="required"/>
        <xsd:attribute name="obdisplayName"
            type="xsd:string" use="required"/>
        <xsd:attribute name="bgcolor" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:binary">
              <xsd:encoding value="hex"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
   </xsd:complexType>
</xsd:element>
<xsd:element name="ObLink">
   <xsd:complexType>
```

```
<xsd:sequence>
   <xsd:element ref="ObCollection"
      minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="appName" use="required">
   <xsd:simpleType>
      <xsd:restriction base="xsd:NMTOKEN">
         <xsd:enumeration value="common"/>
         <xsd:enumeration value=
              "groupservcenter"/>
         <xsd:enumeration value=
              "objservcenter"/>
         <xsd:enumeration value=
              "userservcenter"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="name" use="required">
   <xsd:simpleType>
      <xsd:restriction base="xsd:NMTOKEN">
         <xsd:enumeration value="Admin"/>
         <xsd:enumeration value="Create"/>
         <xsd:enumeration value=
              "FTABconfiguration"/>
         <xsd:enumeration value=
              "FTABcreatereports"/>
         <xsd:enumeration value=
              "FTABorgchart"/>
         <xsd:enumeration value=
              "FTABrequests"/>
         <xsd:enumeration value=
              "FTABviewreports"/>
         <xsd:enumeration value="MyProfile"/>
         <xsd:enumeration value=
              "T1TABgroupmanager"/>
         <xsd:enumeration value=
              "T1TABorgmanager"/>
         <xsd:enumeration value=
              "T1TABusermanager"/>
         <xsd:enumeration value="T1about"/>
         <xsd:enumeration value="T1help"/>
         <xsd:enumeration value="T1logout"/>
         <xsd:enumeration value="Workflow"/>
         <xsd:enumeration value=
              "adminDelegate"/>
         <xsd:enumeration value=
              "adminExpandGroups"/>
         <xsd:enumeration value=
```

```
                    "adminPreWorkflowDef"/>
                  <xsd:enumeration value=
                    "adminProxy"/>
                  <xsd:enumeration value=
                    "adminSetContainmentLimit"/>
                  <xsd:enumeration value=
                    "adminSetSearchbase"/>
                  <xsd:enumeration value=
                    "adminWorkflowDef"/>
                  <xsd:enumeration value="dashline"/>
                  <xsd:enumeration value=
                    "front_page_admin
                      _application_info"/>
                  <xsd:enumeration value=
                    "groupservcenter
                      _application_info"/>
                  <xsd:enumeration value=
                    "multipleObjectTabs"/>
                  <xsd:enumeration value=
                    "objservcenter
                      _application_info"/>
                  <xsd:enumeration value=
                    "userservcenter
                      _application_info"/>
                  <xsd:enumeration value=
                    "policyservcenter
                      _application_info"/>
                  <xsd:enumeration value=
                    "wfCreateProfile"/>
                  <xsd:enumeration value=
                    "wfDeactivateProfile"/>
                  <xsd:enumeration value=
                    "wfIncomingRequest"/>
                  <xsd:enumeration value="wfMonitor"/>
                  <xsd:enumeration value=
                    "wfOutgoingRequest"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="elementName"
                type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="ObNavigation">
    <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="ObHierarchy"
              maxOccurs="unbounded"/>
```

```
        </xsd:sequence>
        <xsd:attribute name="defaultUserType"
          type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Customization

You can make the changes to the `obnavigation.xml` file as described in the list below. The changes take effect the next time the Identity Manager Server is restarted.

To customize the obnavigation.xml file

1. Remove a link.

   To remove access to functionality for a user type, remove the `ObLink` element associated with that functionality. This example shows the original file part revised to remove the `about` functionality for an end user.

   ```
   <ObHierarchy name="oblix" elementName="ObNavbar"
     userType="endUser" obdisplayName="End User"
     bgcolor="CCCC66">
   <ObCollection name="ObMisc">
     <ObLink appName="common" name="T1help"/>
     <ObLink appName="common" name="T1logout"/>
   </ObCollection>
   ```

2. Add a link.

   Use the `ObHierarchy` for the SystemAdmin user type as a template for this. It shows the full standard navigation possibilities. Determine the link to add. Find the `ObCollection` that you would like to add the link to, and add the link. In the revised file part example below, an end user is now able to navigate to the page where new users are created.

   Note: The end user will still need to be granted create rights in order to work with the page.

```
<ObCollection name="ObFunctions">
  <ObLink appName="userservcenter"
       name="MyProfile"/>
  <ObLink appName=”userservcenter”
       name="wfCreateProfile"
  <ObLink appName="userservcenter" name="Workflow">
       <ObCollection name="ObWorkflowFunctions">
```

```
          <ObLink appName="userservcenter"
               name="wfOutgoingRequest"/>
        </ObCollection>
   </ObLink>
</ObCollection>
```

3. Remove a user type.

   Remove all of the `ObHierarchy` elements associated with the user type. That user type will not be able to reach any pages.

   Note: Don't remove the *default* user type. If you must remove the user type that is the default user type, set another user type to be the default.

4. Add a user type.

   Add an `ObHierarchy` element, specifying the new user type. Use the `systemAdmin ObHierarchy` as a template and remove any links and collections not suitable for the new user type.

   Append `&userType=<the user type attribute value in ObHierarchy>` to the entry point URL when you first access the system. The user type information is stored in the cookie that is returned. It will be reset only if a new userType is used in the URL.

5. Set the default user type.

   Change the `ObNavigation deafaultUserType` attribute value to the desired user type. This is used if the user type has not been previously set in a returned cookie and there is no user type specified in the URL.

Valid ObLink Combinations

The following tables show COREid System functionality by application, to be used in defining a valid `ObLink`. For example, if you need to provide the User Manager functionality in User Manager then you would add:

```
<ObLink appname="userservcenter"
          name="T1TABUserManager"/>
```

In the tables, `name` values that are allowed to contain nested `ObCollection` elements are marked with an *.

`Appnames` in the tables correspond to applications this way:

`common`—Help, About, and Logout buttons, and Applications.

`userservcenter`—User Manager.

`groupservcenter`—Group Manager.

Table 1 Valid ObLink name Values for appName=common

name	Description of functionalities common across all NetPoint applications
T1help	Help button
T1about	About button
T1logout	Logout button
userservcenter_application_info	User Manager
groupservcenter_application_info	Group Manager
policyservcenter_application_info	NetPoint Access System
access_front_page_admin_application_info	NetPoint Access System Configuration
front_page_admin_application_info	COREid system configuration
objservcenter_application_info	Organization Manager

Table 2 Valid ObLink `name` Values for `appName=userservcenter`

name	Description of functionalities within User Manager
`T1TABusermanager`	User Manager
`MyProfile`	My Identity
`wfCreateProfile`	Create User Identity
`wfDeactivateProfile`	Deactivated User Identity
`adminProxy`	Substitute Rights
`Workflow *`	Requests
`wfIncomingRequest`	Incoming Requests
`wfOutgoingRequest`	Outgoing Requests
`wfMonitor`	Monitor Requests
`Admin *`	Configuration
`adminAccessControl`	adminAccessControl
`adminDelegate`	Delegate Administration
`adminWorkflowDef`	Workflow Definition
`adminSetSeachbase`	Set Searchbase

Table 3 Valid ObLink `name` Values for `appName=groupservcenter`

name	Description of functionalities within Group Manager
`T1TABgroupmanager`	Group Manager
`MyProfile`	My Groups
`Create`	Create Group
`Workflow *`	Requests
`wfIncomingRequest`	Incoming Requests
`wfOutgoingRequest`	Outgoing Requests
`wfMonitor`	Monitor Requests

Table 3 Valid ObLink `name` Values for `appName=groupservcenter`

name	Description of functionalities within Group Manager
`Admin *`	Configuration
`adminAccessControl`	adminAccessControl
`adminDelegate`	Delegate Administration
`adminPreWorkflowDef`	Workflow Definition
`adminExpandGroups`	Expand Dynamic Groups

Table 4 Valid ObLink `name` Values for `appName=objservcenter`

name	Description of functionalities within Organization Manager
`T1TABorgmanager`	Organization Manager
`multipleObjectTabs`	The set of tabs configured for Organization Manager
`wfCreateProfile`	Create Organization Profile
`FTABrequests *`	Requests
`wfIncomingRequest`	Incoming Requests
`wfOutgoingRequest`	Outgoing Requests
`wfMonitor`	Monitor Requests
`FTABconfiguration *`	Configuration
`adminAccessControl`	Attribute Access Control
`adminDelegate`	Delegate Administration
`adminWorkflowDef`	WorkflowDefinition
`adminSetContainmentLimit`	Container Limits

Index

N

O

P

R

S

T

U

W

X