XLNT®

TECHNICAL DESCRIPTION

ENTERPRISE

COMMAND

AND

SCRIPTING LANGUAGE

FOR WINDOWS

ASCI Advanced Systems Concepts, Inc.

XLNT®

Language is a powerful, and yet easy to use approach to improve Windows System Administration. XLNT's full featured commands, powerful built-in functions (i.e. Lexicals) and easy to use language improve System Administrators and Application Developers productivity by simplifying access, through a Command Line Interface, to all Windows securable objects.

XLNT addresses the needs of System Administrators and Application Developers for:

- ☑ Consistent interface to all Windows securable objects.
- ☑ Powerful Interactive Commands that directly create, modify, delete: Shares, Files, Printers, & Users.
- ☑ Increased productivity, through the incorporation of XLNT commands into VBScript, PERL, and other languages to reduce programming time.
- ☑ Remote System extensibility by allowing XLNT's commands to interact, modify and retrieve information on systems that are not licensed.
- ☑ Reduced development time through the use of our Interactive Development Environment and Debugger for generation of powerful scripts.
- ☑ Faster script development. No compilation or linking required.
- ☑ Protect Intellectual Property Rights and prevent unauthorized script modifications by encoding scripts.
- ☑ Windows Scripting Host (WSH) support by invoking XLNT scriplets.

XLNT Command & Scripting Language

XLNT is an Enterprise Command and Scripting Language featuring interactive commands, built-in functions, and flexible language statements that are surprisingly easy to use. XLNT is a comprehensive solution, that eliminates the need for many third party utilities each with their unique interface, for administering Windows system and network tasks.

WHY XLNT?

XLNT commands are powerful alternatives to the built-in applications that come with Windows XP/2000/NT. XLNT is a comprehensive solution that outperforms many of the Resource Kit utilities and other third party products by providing a lower cost of acquisition and operation through the use of a framework and structured syntax. No longer do System Administrators need to learn the nuances of each Resource Kit Utility to implement solutions such as Logon scripts, for user, group and policy management and other such applications. Each XLNT command provides all of the flexibility found in the native Windows XP/2000/NT utility applications but with improved accessibility directly through the command line.

XLNT gives System Administrators and Application Developers the ability to incorporate each of these commands and functions into sophisticated scripts or command procedures for generating targeted applications for one time or repetitive use.

XLNT SAVES TIME!

The use of “point and click” utilities applications (i.e. User Manager) are effective for today's administrators when creating one or two users or adding a user to a new group. But what happens when an Administrator needs to add 100 users or update the profile of 1500 accounts? The repetitive “point click and type” nature of these utilities is very time consuming and the NET USER command, as an alternative, found in NT’s native command line does not expose nearly as many options and or

XLNT SECURITY Command Syntax

$ SECURITY mode entity-type entity /qualifiers

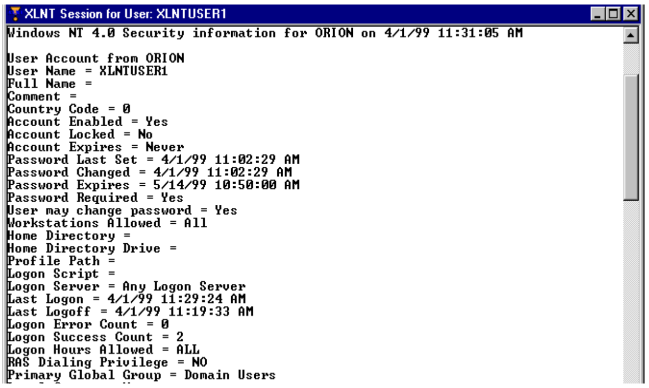
Mode:	CREATE | MODIFY | DELETE | SHOW
Entity-Type:	USER | GROUP
Entity:	User or group name
Qualifiers:	Additional command-line arguments

Qualifiers:

CREATE & MODIFY	DELETE	SHOW
/[NO]ACTIVE	/[NO]CONFIRM	/DOMAIN
/[NO]CHANGE_PASSWORD	/DELETE	/[NO]FORMAT
/COMMENT=string	/GLOBAL	/FULL
/COUNTRY_CODE=nnn	/LOCAL	/[NO]GLOBAL
/CURRENT_PASSWORD=string	/[NO]LOG	/[NO]LOCAL
/DOMAIN[=domain-name]	/ON=machine name	/ON=machine name
/[NO]EXPIRES/[NO]LOG		/OUTPUT[=file-spec]
/GLOBAL[=(GROUP-NAME,…)]		
/HOME_DIRECTORY=file-spec		
/LOCAL[= group-name,…)]		
/[NO]MEMBER=(USER-NAME,…)		
/NAME=string/ON=machinename		
/[NO]PASSWORD[=string](default)		
/[NO]PASSWORD_EXPIRES		
/PROFILE_PATH=file-spec		
/[NO]REQUIRED_PASSWORD		
/RIGHTS=(right-name)		
/SCRIPT_PATH=file-spec		
/[NO]TIMES=(ALL| time-spec,…)		
/USER_COMMENT=string		
/WORKSTATION=(machine-name,…)		

XLNT’s SECURITY Command provides programmatic control for volume operations.

qualifiers (see SECURITY Command Table) as can be found within XLNT. You can manage user accounts easily by issuing a single command or by creating a simple



```
Windows NT 4.0 Security information for ORION on 4/1/99 11:31:05 AM

User Account from ORION
User Name = XLNTUSER1
Full Name =
Comment =
Country Code = 0
Account Enabled = Yes
Account Locked = No
Account Expires = Never
Password Last Set = 4/1/99 11:02:29 AM
Password Changed = 4/1/99 11:02:29 AM
Password Expires = 5/14/99 10:50:00 AM
Password Required = Yes
User may change password = Yes
Workstations Allowed = All
Home Directory =
Home Directory Drive =
Profile Path =
Logon Script =
Logon Server = Any Logon Server
Last Logon = 4/1/99 11:29:24 AM
Last Logoff = 4/1/99 11:19:33 AM
Logon Error Count = 0
Logon Success Count = 2
Logon Hours Allowed = ALL
RAS Dialing Privilege = NO
Primary Global Group = Domain Users
```

XLNT’s SECURITY SHOW USER/Domain command simplifies creating, reading and modify user information

script using our SECURITY command to create, modify, delete or show users or groups. These scripts can then update, in volume or one at a time any or all fields contained in the users account.
XLNT's commands make it easy to create and update User account information, add and configure remote and local printers, set and modify permissions and much more.

XLNT's close integration with the operating system allow Administrators to modify and read the registry, create and modify user accounts, manage files directories and shares on local and remote systems.

XLNT COMMANDS MAKE OTHER LANGUAGES MORE EFFECTIVE

XLNT's powerful commands can also be implemented "right out of the box" within other scripting languages to enhance their native capabilities. Many scripting languages do not have commands that can directly address Windows objects. These languages were often developed and designed to be implemented on other platforms and then ported to Windows. XLNT was designed specifically for Windows and as such its commands can be incorporated into other languages to simplify the process of writing code. Script writers can benefit by making use of XLNT's commands in their existing VBScript, PERL or even .BAT files to lower both the cost of development and maintenance.

REMOTE SYSTEM ADMINISTRATION

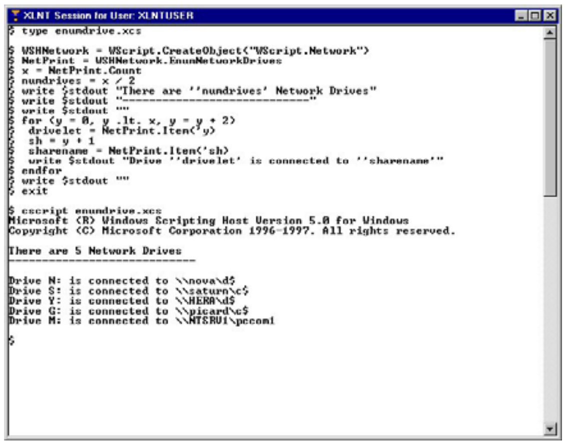
Most of XLNT's commands contain the unique "/ON= machine-name" qualifier so that they can be implemented from the command line on one system to perform a function or read a registry item from another system. The "targeted" system is not required to be licensed with XLNT to perform the operation specified by the command.

As a result of the /ON= qualifier you can start, stop and configure services, check for "free" disk space or add a printer to a remote system from your own console in minutes plus much more. For example to add a printer to a system in another city simply use the following XLNT command:
$ MANAGE ADD PRINTER "TEST PRINTER1" -
"lpt1:" "HP Deskjet 600" /ON=remote node.

Or add the same printer to hundreds of systems by creating a script to identify each of the machines and add "TEST PRINTER1" to every system whether they are down the hall or half way around the world.

WINDOWS SCRIPTING HOST

XLNT supports ActiveX scripting and is a registered scripting engine with the Windows Scripting Host (WSH). As a result you can combine the best of object manipulation with our powerful XLNT commands. As an example, you could open an Excel spreadsheet, populate the spreadsheet with User Account information

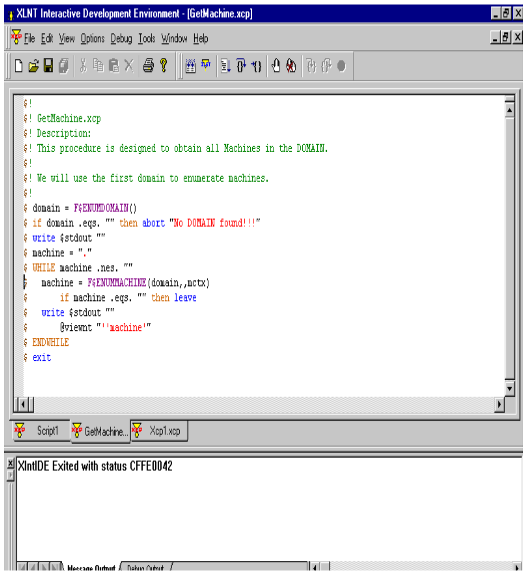


```
$ type enumdrive.xcs

$ WSHNetwork = WScript.CreateObject("WScript.Network")
$ NetPrint = WSHNetwork.EnumNetworkDrives
$ x = NetPrint.Count
$ nundrives = x / 2
$ write $stdout "There are ''nundrives' Network Drives"
$ write $stdout "-----------------------------"
$ write $stdout ""
$ for (y = 0, y .lt. x, y = y + 2)
$  drivelet = NetPrint.Item('y)
$  sh = y + 1
$  sharename = NetPrint.Item('sh)
$  write $stdout "Drive ''drivelet' is connected to ''sharename'"
$ endfor
$ write $stdout ""
$ exit

$ cscript enumdrive.xcs
Microsoft (R) Windows Scripting Host Version 5.0 for Windows
Copyright (C) Microsoft Corporation 1996-1997. All rights reserved.

There are 5 Network Drives
-----------------------------

Drive N: is connected to \\nova\d$
Drive S: is connected to \\saturn\c$
Drive Y: is connected to \\HERA\d$
Drive G: is connected to \\picard\c$
Drive M: is connected to \\NTSRV1\pccom1

$
```

and then add, modify or use that information for another function.

INTERACTIVE DEVELOPMENT ENVIRONMENT

XLNT's Professional Edition includes a specialized Interactive Development Environment and Debugger. The IDE provides a single point in which the Script Writer can create, maintain and debug their scripts. It is language sensitive and provides professional development capabilities.


XLNT Interactive Development Environment - [GetMachine.xcp]
File Edit View Options Debug Tools Window Help

```
$!
$! GetMachine.xcp
$! Description:
$! This procedure is designed to obtain all Machines in the DOMAIN.
$!
$! We will use the first domain to enumerate machines.
$!
$ domain = F$ENUMDOMAIN()
$ if domain .eqs. "" then abort "No DOMAIN found!!!"
$ write $stdout ""
$ machine = "."
$ WHILE machine .nes. ""
$   machine = F$ENUMMACHINE(domain,,mctx)
$       if machine .eqs. "" then leave
$   write $stdout ""
$       @viewmt "''machine'"
$ ENDWHILE
$ exit
```

Script1
GetMachine...
Xcp1.xcp

```
XlntIDE Exited with status CFFE0042
```

The IDE and Debugger make it easy to develop and debug even the most complex scripts.

As you develop your scripts you can test them from within the IDE itself which provides complete debugging facilities. Scripts can be executed or you can elect to step through using the debugging commands.

XLNT scripts can be "compiled" or "encoded" using the Professional Edition. By creating executable images you can protect your Intellectual Property Rights, or prevent unauthorized changes by users to deployed scripts.

CLUSTER MANAGEMENT IS SIMPLIFIED WITH XLNT!

Security is, of course, of paramount importance to the administrators of Windows systems. The XLNT *Set Permissions* command provides a simple, flexible method of securing your systems' assets. It enables you to explicitly grant or deny access to any securable object. Securable objects may be files, directories, shares, registry keys, printers, or services.

With the *Set Permissions* command, you specify the object to be secured, and the entities (users, groups, etc) that may access it, and how they may access it. For example:

```
SET PERMISSIONS C:\TEST -
/ACCOUNT=(JSMITH:FULL, GUESTS:READ)
```

This command grants user JSMITH full access to the C:\TEST directory, but members of the GUESTS group are give read-only access to the directory. As can be seen by this example, the */ACCOUNT* qualifier is used to specify the entities that can access the securable object. The *name* portion of this qualifier represents the name of the account or group that is being granted or denied access to the object. It may be specified as *machine\name* or *domain\name* or simply as *name*. You may also specify this value as a SID (Security Identifier) using textual SID notation (*S-n-n-n-n…*). This latter is especially valuable within a cluster environment, where other members of the cluster may not recognize account names but where the SID is known. As a result time is saved, reliability improved by reproducing the same operation across all cluster members with XLNT.

SUMMARY

XLNT is a comprehensive solution that eliminates the need for many third party utilities and their disparate interfaces, for administering Windows system and network tasks.

ASCI Advanced Systems Concepts, Inc.