Inno Tech

Consulting Group LLC

Universal Login Script

Configuration Guide

Version 2.8.7
2014/10/10

KiXtart Universal Login Script – Version 2.8.7

Trademarks

Copyright

Restricted Use

Acknowledgements

Thanks to the following people for their support, feedback, and suggestions for making this product the best it can be:

Tony Kwan, Usul, Stephen Burke, Vincent, and all the rest of you that took the time to provide feedback and suggestions over the years…

Contents

Introduction

The *Universal Login Script* was developed with the original release of 16-bit KiXtart in 1995. It's called "universal" because it has been used at dozens of companies without the need for code modification since 1996. Of course, there have been enhancements over time to take advantage of new Kixtart functionality, and enhanced capabilities, but the core concepts of tight code and flexible operation have remained.

During 2006, the code was completely rewritten to meet our stringent coding standards. The original logic was preserved, but the script itself was made more modular and the subroutines were converted to UDFs. The INI file format was also updated to use a more flexible format of resource records.

With over 18 years of deployment history in organizations both large and small, the *Universal Login Script* can accommodate almost any situation. It offers sophisticated capabilities to authorize access; powerful data rewriting to increase flexibility; and extensive logging and debugging features to aid in implementation and support. It can display messages in multiple languages, even adding new languages in just minutes. It supports custom messages or even no display at all, can process drive and printer connections, display network messages, and run commands all without writing a single line of code.

Some key capabilities of the *Universal Login Script* include

- Authorize access to resources by user privilege type, Active Directory Group or OU membership, Computer OU membership, AD Attribute, or custom-coded attributes such as workstation name.
- Connect to disk and printer resources.
- Display text messages with user-definable delays.
- Run commands – even other Kixtart scripts that can use the functions in the core script! Commands can be run synchronously or asynchronously with the login process for utmost flexibility.
- Alter the server, share, or path of a resource based on UserID, User OU, Computer OU, Site, or network subnet via Value Rewriting technology.
- Control mapping based on Desktop or Laptop computer types or console, RDP, or Citrix (ICA) logon sessions.
- Config files can be defined per site, computer, or even per user for maximum flexibility.
- Multiple configuration sets can be defined in a single configuration file, allowing both common and site or region-specific settings to be maintained centrally.
- Alternate-path mapping for disk resources for DR resiliency.
- "Code-free" configuration – no scripting knowledge is necessary to deploy a sophisticated login script. Simply define Resource Records, test, and deploy.
- Display messages in alternate languages, including user-defined messages.
- Extensive documentation and examples, and a GUI Admin Console.
- Free and Paid support options get you up and running quickly and keep you running.

A login script is the one administrative tool that touches every user on your network. It should be reliable, flexible, and robust, all while being unobtrusive. This is exactly what the *Universal Login Script* provides.

Overview

The *Universal Login Script* package consists of three core files - the *Kix32.exe* script interpreter, the *Kixtart.kix* login script, and the *login.ini* configuration file. All of these files must reside in the same folder within the netlogon share, where they can be replicated to all other Domain Controllers in your environment.

The script also supports several optional files that extend the capabilities of the core script. These optional files include one or more *LSL_###.LNG* to provide custom language message files; the *UserProcess.kix* script to perform custom authorization processing, and the *UserDisplay.kix* script to display a custom welcome screen. Additional configuration files may be present as well.

Theory of Operation

Here is the basic theory of operation:

- The script is invoked by running *Kix32.exe*. The script engine looks for *Kixtart.kix* in the folder where it was started.
- The login script looks in the startup (netlogon) folder first for a configuration file. It will look for a common file, a file specified on the command-line, a group-based file, a computer specific file, and then a user-specific file in that order.
- The script checks the user's TEMP folder for a copy of the config file. If it is not present or is different than the one on the server, the server's config file is copied to the local computer, effectively caching the configuration.
- Operational parameters such as the preferred message language, connection type, domain name restrictions, and message suppression are loaded from the config file.
- If present, *UserProcess.kix* is loaded and initialized.
- Additional language files are loaded to display messages in the user's language.
- If messages are not suppressed, the welcome screen is displayed using either internal language-specific messages or the optional, external *UserDisplay.kix* script.
- If enabled, previous disk and printer resources are disconnected.
- The entire config file is parsed. The resource records are evaluated to determine if the user is authorized to access them. Any unauthorized record is ignored, and the rest are queued for processing. Note that the login script determines if a user is authorized to *connect* to the resource, not actually *access* the data it contains.
- The authorized resources are processed – drives are mapped; printers are connected; messages are displayed; and commands are run. If messages are authorized, they will be displayed even if output is suppressed. If a drive resource is associated with a drive letter that is already mapped (eg: persistent connection), the prior mapping is removed and the new path is mapped. This behavior is not configurable. Also note that if multiple disk resources are mapped to the same drive letter, the last connection will prevail unless connection priorities are defined.
- If any connection or processing errors have occurred, a message is displayed recommending a call to the help desk, along with the error messages. These error messages are displayed even if message display is suppressed.

Special Capabilities & Features

There are several special capabilities that the *Universal Login Script* offers that are not found in other login scripts or would take significant effort to code into your own script. These features might improve performance, extend the usability in larger environments, or help identify process and configuration issues.

Config-File Caching

The *login.ini* file is cached on the local user's computer to improve processing of resources, particularly across WAN links. Once the script determines whether a user or system config file will be used, it checks the local computer for a copy of that file. If the local file does not exist, or is different than the one on the server, the file is copied from the server to the local system, where it is used directly until it needs to be updated again. Note that any modification of a local config file will cause the server copy to be downloaded when the script is invoked. This prevents users from running an unauthorized configuration.

Multiple Config-File Selection Methods

A user-specific configuration file is associated with a User ID and is the highest priority for configuration file selection

A configuration file can be associated with a specific computer and is the second-highest priority. This method is useful for special purpose computers or Remote Desktop/Citrix environments where a configuration must be modified or replaced. Computer-specific files are *merged* with the primary configuration file by default but can be configured to replace it completely. In merge mode, resources in the computer-specific file have priority.

A configuration file can be specified on the command-line.

Group-based selection allows a configuration file to be selected based on a user's membership in an AD group. This can provide a more dynamic solution than specifying a config file name or configuration set name on the command line. Due to extra overhead this method is designed primarily for temporary use during login script migrations.

Configuration Sets

A configuration set (CFGSET) allows a single config file to contain resources common to all users as well as resources relative to a specific collection of users. This would be useful to connect a resource based on an application that a user requires, possibly overriding other resources matched by Group or OU attributes.

Internationalization

The *Universal Login Script* natively supports message display in five languages, and can support additional languages through the use of .LNG files. User messages are automatically displayed in the native language (where available) based on the user's LOCALE setting, and default to English if a language is not available. Debug and Log messages are always written in English.

AD Attribute-Based Authorization

Resources can be authorized by AD Attribute, allowing resources to be connected based on attribute values such as Department Name, City, or any other AD/LDAP attribute.

User-Defined Resource Processing

Customized resource processing can provide environment-specific methods of resource authorization. As each resource record is processed, all standard authorization evaluations are performed. Then, if the user-defined function is active, it is called so custom evaluations can be processed. Only one function is available, but it can be written to provide different functionality depending on the resource class, and even adjust based on custom resource values. This provides a high degree of customized control with a minimum of programming.

One application for the user-defined process is machine-based processing. In a school, for example, computers might need classroom-specific resources. Those resources can now be bound to the machine instead of a user by granting access based on computer name or name fragment if the computer name includes the room ID.

Value Rewriting

Value Rewriting allows a macro to be placed in a UNC path or text description, allowing that value to be dynamically modified based on a broad set of parameters. This feature can reduce the number of resource records and would eliminate hundreds of “If” statements from the script if you wrote it from scratch. Rewriting can be done based on User, Computer hostname, OU, Computer OU, IP Subnet, and AD Attribute, and support both direct replacement and table-based lookups of replacement values.

Secondary Drive Paths (D-R / B-C support)

The path to a disk resource defined in the config file can actually contain multiple paths delimited with a semicolon. During normal operation, the connection is made to the first path. If the primary connection fails, the additional paths are tried until one succeeds. If no path connections are made, an error message is displayed. A simple solution for disaster-recovery and business continuity operations.

Local Default Printer

The ability to set a locally attached printer as the default is available by defining a local environment variable. Since the login script does not “map” this local printer, there is no resource record associated with it. The computer must employ a locally-defined System Environment variable called “DPRINTER”, and it must contain the exact printer name to be set as the default. This feature is especially useful when a user has a local printer for security or confidentiality reasons. It can also be used to override the selection of network printers, as this parameter has the highest priority when defining a printer’s default status.

Diagnostic Logging

In-depth logging is available for testing, troubleshooting, and performance analysis.

A debug file – *LoginDebug.log* – records the results of all procedures related to parsing, authorizing, and accessing of resources. This is instrumental in validating complex authorization parameters in the configuration file.

An error log – *LoginErrs.log* – is created automatically whenever a login script detects an error connecting to a resource. This log lists the date & time, the process being run (ie: Drive Map D: to \\server\share), and the resulting error message.

A performance log – *LoginPerf.log* – records timestamps as the script runs with 15-millisecond accuracy. This helps identify bottlenecks in the configuration, problems with network resources, and slow-running external commands.

Licensing

The Universal Login Script is licensed for unrestricted use in Active Directory domains with up to 4 domain controllers. In environments with more than 4 Domain Controllers, an enterprise license must be purchased. Connect to http://www.innotechcg.com/register to purchase a license pack. Purchases can be paid through most major credit cards or *PayPal*. Corporate purchase orders are accepted from most major corporations and educational institutions. Send an email to *sales@innotechcg.com* with your purchase request.

Current pricing can be found on the registration page. We strive to make our products cost effective for environments of any size. Licensed users get 2 additional support instances and priority response to support requests for the year following the purchase.

Licensing fees are based on a core of 5 licenses and then expansion packs of 5 additional licenses. Discounts are applied to volume purchases of expansion packs up to an unrestricted license for 50 or more domain controllers in a single domain.

The License Process

To obtain a license key, generate a license key request by running

```
      kix32.exe kixtart.kix --lkr
```

from a command prompt in the directory where the script is located. The script will generate a License Key Request string consisting of three sets of 8 digits. It will also display the number of licenses required for your environment (in multiples of 5). If you are near or at the required number of licenses, the screen will display a message recommending an additional license pack be purchased.

Browse to the registration URL shown above and enter the License Key Request string. After entering the string, the display updates to allow selection of the minimum or desired number of licenses. Enter your email address – it is important that this be entered correctly as the license key will be sent automatically to this email address after payment is authorized. After verifying the information, you will be brought to a secure screen where you can enter credit card or *PayPal* data to complete your purchase. Upon validation, the license key will be emailed, usually within 60 minutes.

If you do not receive your license key within a reasonable time, forward a copy of the License Key Request string and your payment validation to *support@innotechcg.com*.

You will receive a license key file called LoginScript.lic – place this file in the same location as the Kixtart script. If you use multiple folders to manage login script configurations, place a copy of the license file in each folder. Licenses are domain wide but must be located in the same folder that the script starts from. The key can also be placed into the config.ini file.

Unlicensed Operation & Testing

In environments where a license is needed for operation, the script will run in debug mode during logon, writing all events to a log file. For functional testing, the script can be invoked manually after logon and will function normally, mapping all resources and performing all functions. This allows complete capability for testing and operational validation.

Installation, Configuration, & Testing

Installing the login script is quite simple, and begins by using an off-line location to develop and test your configuration file.

1. Start by extracting the distribution files to an empty development directory. Place a copy of Kix32.exe (downloaded separately) into the same directory.
2. Copy the appropriate compiled script to Kixtart.kix. The scripts extracted from the ZIP are named Kixtart_*x.xx*.KIX, representing the version of Kixtart that the script was compiled with. The same source code is used, but minor differences in the compiled output require version-specific files to be produced.
3. Make sure that a DEBUG.TXT file exists in the development folder. This will force the login script to run in a special (Level 2) debug mode, which will only display the resource names and not actually process them. It will also force the script to use the login.ini file from the current directory, where you would be developing the configuration file. Normally, the login.ini file will be loaded from the same folder as where Kix32.exe was invoked. This also prevents the use of any cached copy of the configuration file.
4. Review the documentation related to defining resource records. Examine the sample login.ini file to gain an understanding of how the resource records work. You can run the KIX32 command right after extraction and the sample configuration file will be parsed in debug mode. Very few of the resources will be authorized because the sample configuration has generic Group and OU names. Change some of the parameters to use names valid for your environment for a better demonstration.
5. Create a new *login.ini* file using the sample provided and edit it to meet your requirements. Much of the COMMON section can be used as-is or with minor parameter changes. Launch *KIX32.exe* to run the script and review the mappings and processes that would be performed. When you are satisfied that your configuration file is working properly, you can continue with the installation process in your AD environment.
    a. If you plan to use the *UserDisplay.kix* to customize the initial greeting screen, review the section titled "*Customizing the Initial Display Screen*" later in this document. Create/edit the script and include it with your other files.
    b. Develop & test the *UserProcess.kix* script if you plan on using it. Refer to the section titled "*Custom Resource Processing*" for more details.
    c. If you need custom language support, create one or more *LSL_###.LNG* files to define the messages in the language(s) of your choice. See the "*Customizing Language-Specific Messages*" section for full details.
6. Copy the files to your domain controller's SYSVOL\domain\scripts folder, from where they will be replicated to other domain controllers. You will need the *Kix32.exe*, *kixtart.kix*, your customized *login.ini*, and – optionally – the *login.bat* file. Login.bat is needed only if you support legacy (Win9x) clients. You will also need to copy the *UserDisplay.kix*, *LSL_###.LNG,* and *UserProcess.kix* files if you use them.

7. Using a test account, define the account properties to run the new login script. In most environments, all you need to do is reference *kix32.exe*. Note that Kixtart 4.60 has a bug that requires you to specify the login script name – “kix32.exe kixtart.kix”.
8. To validate the process, login with a normal (not test) account on a test system and create an empty file – *LoginDebug.log* – in the test user’s profile folder (%USERPROFILE%). Log off, and then log back on with the test account. The presence of the debug file will be detected, and status information will be written to that file. Verify that the appropriate devices were mapped, message files displayed, and extra scripts executed. Note that in this debugging mode, resources are mapped, displayed, and executed as appropriate, and their action is logged to the log file. After verifying your configuration, you can deploy the login script to the remainder of your user population. Be sure to remove the LoginDebug.log file from your test system when you are finished debugging your script configuration.

   Note: *As of version 2.7.5, you can add the parameter “`--D`” to the Kix32.exe command in the user profile to force debug mode on startup, eliminating the need to log in and create the log file manually.*

Note that any time you need to debug your script configuration, you can create the *LoginDebug.log* file in your UserProfile folder. The mere presence of that file will cause the debug information to be written. The file is overwritten each time the script runs to prevent it from growing uncontrollably. A typical debug file is only 5-7 Kbytes.

You can also enable debugging system-wide by setting a DEBUG=Y parameter in the COMMON section of your config file. This is not generally recommended as it will enable debug logging for all users and can affect performance for everyone by creating a debug file in every user’s profile folder. Since the presence of that file itself enables debugging, turning this on just once will enable debugging for every user’s logon until the local file is deleted from every user’s profile folder AND the setting removed from the configuration file! *This setting, therefore, should be used with extreme caution and only in test environments!*

Pre-Deployment Considerations

System Policy Settings

We strongly recommend that any logon script be run visibly so the user is aware of any connection or processing errors, and run synchronously so that all mapped resources are available before the user begins working. The script runs extremely quickly so synchronous operation will not cause unacceptable logon delays. Enabling the setting to “Wait for the network to be ready” is also a recommended practice, especially in large networks or those with multiple sites.

Login (profile) Vs. Logon (GPO) Scripts

The *Universal Login Script* is designed to function as a Login script defined in the User Profile. This script runs in the user context and can properly configure all user settings. Logon scripts deployed via GPO run in a system context and may not be able to properly process all user-specific settings. The use of the Login script method applied via User Profile is strongly recommended to insure the highest compatibility of all features.

The Login Configuration File

The configuration file consists of three types of records.

- The first type of record – the *common* record – defines the overall script configuration parameters, and is defined in the COMMON section.
- The second type is a *Resource* record, which defines network resources that can be connected to, displayed, or executed. There are many resource records, usually one for each resource configuration to be processed. Realize that there could be multiple records for one drive letter, or for one share, depending on the authorization qualifications that you define. There could also be one record for many resources through the use of Value Rewriting and Lookup capabilities.
- The third type is a *Lookup* record. This is used by Value Rewriting to translate a user name, site name, OU string, or network address into some part of the resource value. You may have as many lookup records as you need. There is also a special Lookup record called *DEFAULTPRINTER*, which allows overriding the default printer selection on a per-computer basis, and another called *GROUP_BASED_CONFIG* for selection of configuration files by AD Group membership.

All records are maintained in *login.ini*, which is a standard INI format file. Each section will be reviewed in detail in the pages that follow. The records can be defined in any order, although the COMMON section is usually placed first. The sample configuration file contains many comments that can remain or be removed. Having comments in the file will not affect the operational performance.

Locating the Configuration File

The *login.ini* configuration file is located by determining which folder Kix32 was invoked from. This provides a simple yet effective method of providing different configuration files within a single Active Directory domain. For example, there might be three unique divisions within the Fabricam.com domain – Widgets, Gadgets, and Thingamajigs. Each division has its own servers, but all of them are all in the same domain. The login configuration can be simplified considerably by creating a subfolder for each division. The NetLogon share would have a Widgets, Gadgets, and Thingamajigs folder. Each folder would contain a copy of *Kix32.exe*, the login script *Kixtart.Kix*, and a unique *Login.INI* file. User profiles would specify the path to kix32. For example, instead of simply placing “kix32.exe” in the login script field of the user’s profile, users in the Widgets division would have “widgets\kix32.exe”, while users in the Gadgets division would have “gadgets\kix32.exe”.

Alternate Config-File Selection Methods

The *login.ini* file is the default configuration file for all users. There are several alternate methods of choosing a different configuration file to accommodate special situations. The alternate files are processed in a specific order of increasing preference as listed here.

Selection by Command Line Argument

An alternate configuration file can be specified on the command line with `--i` *`name`*. The .INI extension is not specified. This method is useful when you want to run site or division-specific configuration files from a common NetLogon folder.

The use of the command line to specify a config file is equivalent to using a subfolder with the standard login.ini file, and the decision to use one method or the other is left to the local administrator. For example, specifying `kix32.exe kixtart.kix --i widgets` would have the same effect as using a widgets subfolder in the NetLogon folder and specifying `widgets\kix32.exe kixtart.kix`.

Selection by Group Membership

When enabled, a configuration file may be selected based on a user's membership in a list of defined AD Groups. This is discussed in detail in the section titled "Group Based Config-File Selection".

Selection by Client Computer

A computer-specific configuration file can be defined using the format *hostname.ini*. When a user logs into a computer and a corresponding *hostname.ini* file exists in the NetLogon folder, that file will be used according to the following rules:

- If the *MachineSpecificReplaceMode* value in the *hostname.ini* file is enabled, or the "`--omc`" command-line argument is specified, the *hostname.ini* file will replace any previously selected configuration file (replace mode).
- If replace mode is not enabled, the *hostname.ini* file will be processed after the default configuration file and the contents merged. Any resource defined in the machine-specific file will override any resource that is also defined in the primary configuration file.

Computer-specific configuration files are an ideal way to customize resources for special hosts such as Kiosks and Instructor systems in classrooms. They are also an excellent method to alter the configuration for application-specific Remote Desktop/Citrix servers.

Selection by User ID

If the login script finds a *userid.ini* file, it will use that file for the mapping the user's resources. This feature should be used only when all other configuration options are exhausted, as the maintenance of individual configuration files can be overwhelming. It is, however, an excellent way for one user (you!) to test a new configuration file prior to deploying it to the user community!

The COMMON Configuration Parameter Section

These parameters are contained in the COMMON section of the configuration file. All of these values are optional, and control general operation of the login script.

- **License Key** – *optional* – A string containing the license key, where required.
- **HDMessage** – *optional* – A string displayed when errors are encountered, informing the user how to contact the help desk. This message should be relatively short, and should include contact info such as a phone number or extension.
- **ConnType** – *optional* – A comma-delimited list that defines the permitted connection types – CON (Console), RDP (Terminal Server), or ICA (Citrix**).** If this record is defined, only connection types that match *one* of the entries in this comma-delimited list will run the login script. For example, if you do not want users of your Citrix server to run this login script, specify "CONNTYPE=CON,RDP", excluding the Citrix ICA connection class. You can also specify the connection type restriction on a resource by resource basis. If ConnType values are specified for the common parameter and the connection type does not match, the script will exit silently without any messages being displayed. If the string is blank, then no restrictions are enforced.
- **ClearDriveMappings** – *optional* – A Boolean value that determines if the script should clear persistent drive mappings before mapping drives. Specific drives can be excluded by defining the IgnoreDriveMappings parameter. This parameter (and all other Boolean parameters) can be defined as "Y" or "N".
- **IgnoreDriveMappings** – *optional* – A string of drive letters that should not be unmapped when ClearDriveMappings is true. This is useful when a range of drive letters is permitted for persistent ad-hoc mapping. The drive letters should be delimited with commas.
- **ClearNetShorcuts** – *optional* – A Boolean value that determines if the script should remove network folder shortcuts from Network Neighborhood.
- **IgnoreNetShortcuts** – *optional* – A string of either UNC Paths or Shortcut Names that should not be removed when ClearNetShortcuts is enabled.
- **ClearPrinterMappings** – *optional* – A Boolean value that determines if all printer mappings should be cleared before mapping printers.
- **ForceVisible** – *optional* – A Boolean value that forces the login script to run in a visible, maximized window. This setting overrides the Group Policy "run login script minimized" by forcing the console to be a tall, foreground window.
- **MinimumDisplayTime** – *optional* – Force the login script window to display for the defined number of seconds. Useful in very fast environments – especially virtual systems – where there is insufficient time to view the status information that is displayed during login. The maximum allowed value is 10 (seconds). Any value specified that is greater than 10 is forced to 10. This value is measured from when the script starts running, so if the script takes 11 seconds to complete, no additional delay will be added. If you desire an additional display time, which might be useful for

debugging, you can create a Kix script that sleeps for the additional seconds that you desire or waits for a keypress, and CALL it via the last COMMAND record.

- **FlushTokenCache** – *optional* – A numeric value defining the number of days that can elapse before the Kix32 token cache is flushed. The cache-flush takes effect on the *next* login, since Kix32 is already running this process.
- **OUOffset** – *optional* – A value that defines the element in the DN string that uniquely identifies the unique department or divisional OU. See Path Rewriting for more information. The value defaults to 1, assuming that a user DN string looks something like:

  CN=user name,OU=Users,OU=Dept,DC=contosso,DC=com

  In this case, the desired OU contains "Dept", and is the second OU element. Using a zero base, that element has an ordinal of 1.
- **PCOUOffset –** *optional* – Same as OUOffset, but for the workstation name
- **LanguageID** – *optional* – Defines the language for user messages and disables automatic language configuration. If the value is 0 or undefined, the script will attempt to display messages in the language defined by the user's Locale setting, defaulting to English if a language is not available.

  See the appendix for a table of Locale values that can be set. Not all Locale languages are available, but language files can be generated at no charge if a translation is provided. You can also create your own language file using the UserLang.kix file as a template for a LSL_###.LNG file. ("###" is the Locale ID.)

  Up to ten core languages are built into the script. Additional language files are loaded dynamically when needed. The messages are displayed in English when a required language is unavailable. See the readme.txt file for the most current list of supported languages.

  The LanguageID value may also contain Rewrite:Lookup values. Instead of a specific numeric value, the parameter can be one of the following:
  - "&SITE:LanguageTable&" to perform AD-Site-specific language settings
  - "&OU:LanguageTable&" for OU-specific language settings
  - "&SUBNET:LanguageTable&" for subnet-specific language settings

  A Lookup record matching the referenced name ("LanguageTable" in the example above) must be created with the values to associate the Site, OU, or Subnet to specific language IDs.

  It is recommended to allow dynamic language selection whenever possible by setting this value to zero (or undefined).
- **RunSilent** – *optional* – Suppresses all script output except for MESSAGE records and error messages. Causes the ForceVisible option to be ignored. Note that if Group Policy is set to run the login script minimized, any system messages may not be seen at all if RunSilent is selected. Messages and errors are not suppressed by RunSilent.

Note that using GPO to run the script minimized is not recommended as errors will be displayed but unseen in the minimized window.

- **LPHardMap** – *optional* – When true, remaps printers that are already connected. The default is soft-mapping, which will not remap a printer that is already connected.
- **AllowedDomains / DeniedDomains** – *optional* – lists of domains where the login script is permitted or denied. This is useful where inter-domain trusts exist for authentication but access to "home domain" resources may be blocked. The login script will simply abort with an "invalid domain – script terminated" message after the initial user information is displayed. The MinimimDisplayTime parameter is honored, insuring that the user can understand why the script terminated. *Only one parameter may be used – either a list of allowed domains or a list of denied domains!* This is not required when you have multiple domains, but may be helpful when logging into trusted domains.
- **NAC Settings** – *optional* – controls the actions of the script when Network Access Control is active (the netmask is 255.255.255.255).
  - **NAC_Enable –** *Boolean* – enables NAC awareness.
  - **NAC_Timeout** – *Integer* – number of seconds to delay during NAC condition.
  - **NAC_Command** – String – command to run after delay
  - **NAC_Command_Delay** – Integer – Seconds to delay before running command.
- **UseGroupBasedConfig** – *optional* – A Boolean value that enables the use of configuration file selection based on Active Directory group membership. When true, the GROUP_BASED_CONFIG section of the INI file is enumerated to determine if the user is a member of one of the specified groups. If a membership match is found and the defined configuration file exists, it is used for all further configuration tasks. If no group membership match is found or the defined config file does not exist, the default configuration file is used.
- **MachineSpecificReplaceMode** – *optional* – A Boolean value used only in a machine-specific configuration file to indicate that the contents should replace rather than be merged with the primary configuration file.

The GROUP_BASED_CONFIG Section

This is a lookup section that maps Active Directory group names to config file names, using the format “AD_GroupName=Config_Filename”. This is used only when the UseGroupBasedConfig setting is true. Use is discouraged except during migration tasks due to the extra processing required.

The configuration of this section is discussed in detail in the User Guide section titled “Group Based Config-File Selection”.

The DEFAULTPRINTER Section

This section is a lookup table, but is not associated with any Value Rewrite parameter. Its purpose is to allow the centralized management of per-computer printer default settings. In concept, it works similarly to the DPRINTER environment variable to define a default printer, but without the need to configure each computer.

Note that this setting cannot be used to define a locally attached printer as the default – the use of the DPRINTER environment variable is still required for this purpose.

The format of this section is simply <ComputerName>=<PrinterResource>, where the computer name is the standard NetBIOS computer name and PrinterResource is the full UNC path to the printer resource.

The general concept for use of this feature is to define one or more printer resources. These resources can set the Default Eligible status as needed, allowing the printer to become the default printer for most systems. A small group of computers can then override the default printer setting by creating an entry in the DEFAULTPRINTER lookup table.

DEFAULTPRINTER is used to set a *network* printer to default, overriding any default selection. The DPRINTER variable defined in the computer environment is used to override the default printer selection with a *local* printer.

Resource Records

Resource Records define all of the parameters associated with a network resource. The parameters include classification, location, description, and an array of authorization values. A record might control one or many possible resource targets through Value Rewriting.

There are four basic types (classes) of resource records found in the configuration file. These include *Disk*, *Print*, *Message*, and *Command* records. Disk records map to network disk resources. Print records are used to connect to network printer resources. Message records are used to display the contents of text files, usually for outage notifications or other network events. Command records execute other processes, which can install or remove software, or customize the target computer. Note that commands are executed in the user's security context. If a user is not a member of the local Administrators group, many commands and installations may not be successful. We recommend that the command resources be limited to processes that customize the user's workstation environment.

Resource Record Format

All resource records follow a standard format beginning with a Resource Identifier and followed by two or more parameter/value pairs. CLASS and PATH parameters are always required. A typical Disk resource record is shown here:

```
[RESOURCE_ID]
CLASS=DISK
PATH=\\server\share\subfolder
TARGET=S:
DESC=Share identifier string
GROUPS=group1,group2,Group3!
```

This will map the shared disk resource to the user's S: drive if the user is a member of group1 or group2. The user cannot be a member of group3, however. The mapped drive will be labeled as "Share identifier string", which will be visible in Windows Explorer.

It is critical that each Resource ID be unique! If duplicate Resource IDs are present in the *config.ini* file, any entries after the first will be ignored.

Note that different classes of Resource Records have different number of required parameters. For example, DISK records require CLASS, PATH, and TARGET parameters, while PRINT records only require CLASS and PATH parameters.

```
[RESOURCE_ID]
CLASS=PRINT
PATH=\\server\Printer23
DESC=high speed laser printer
OUS=Corporate HQ
SETDEFAULT=y
```

This example will map the high speed laser printer to all users in the "Corporate HQ" OU and set it as the default printer. The DESC parameter is not used by the login script and is present only for identification purposes.

Resource Core Parameters

Resource records consist of the following parameters that define the type and location of the resource. The content of each value may differ based on the class of resource being defined.

- **CLASS** – *required* – This defines the type of resource record.
  ```
  CLASS=Disk
  ```
  Values must be one of:
  - **DISK –** defines a shared disk resource.
  - **PRINT –** defines a shared printer resource.
  - **MESSAGE –** defines a message resource file that will be displayed.
  - **COMMAND** – defines a command resource file that will be executed.
- **PATH** – *required* – Defines the path to the remote resource. Path Rewriting is supported for all resource classes. Paths for all resource types may include environment variables, which are expanded before use.

  Message path values have a special feature that auto-locates files within the NetLogon share. If the path is rooted – beginning with a "\\" or "*d*:\", the actual path is used. If the path is unrooted, the \\%USERDOMAIN%\NetLogon share (and the startup subfolder, if different) is checked instead. If the file is found there, the path value is updated to define the location within the NetLogon share. File names can be prefaced with an unrooted path to define subfolders within the NetLogon share as shown here.
  ```
  PATH=folder\filename
  ```
  - **Disk** – A UNC path referencing the network disk resource, usually in a "\\server\share" format, but it can also include additional path values for deep-mapping as supported on Windows 2000 and higher environments. Multiple paths can be specified, delimited with semicolons. If one path is not found, the others are tried before reporting an error, which is useful for D-R scenarios. This value can also be the literal string "HOME", which refers to the user's home folder as defined in Active Directory. The HOME mapping is ignored by 32/64-bit systems in AD domains (where it is mapped automatically) but can be defined so a mapping reference is displayed in the login script console. If the login script is run outside of the login process (manually after a VPN connection, for example) the HOME value will be looked up and mapped via the login script if it is defined in the user's profile. The HOME parameter is ignored if not defined in the user's Active Directory profile.
  - **Print** – The UNC path referencing the network printer resource. Note that the printer name and share name must be identical for proper printer connections.
  - **Message** – The path to the message text file to display in the login script console window.
  - **Command** – The path to the command file to run, without any arguments. The exact path must be specified if the command cannot be located via the System PATH variable.

- **TARGET –** *class-specific* – Defines the target identifier that the resource will be mapped to.
`TARGET=G`
    - **Disk –** *required* – The local drive letter, with or without a trailing colon. It can also be the string "UNC", which will create a network shortcut in Network Neighborhood instead of mapping to a drive letter.
    - **Print** – *optional* – The local LPT# device name (rarely used), used to support DOS/Legacy applications. LPT1-LPT9 can be defined.
- **DESC** – *optional* – Defines a description for this resource. Value Rewriting is supported with both direct and lookup based rewrites.
`DESC=description of resource`
    - **Disk** – On Windows XP and higher systems, the mapped share will display this description in Windows Explorer. As such, it should usually be kept short - 30 characters is a good maximum length.
    - **Print –** not used, but allowed as a comment.
    - **Message –** A small text header that precedes the actual file content
    - **Command** – If specified, the message "Please wait while <DESC> runs" is displayed. If DESC is not specified, the entire command path is used in the "please wait" message. The DESC parameter is recommended to provide a more meaningful message to the end-users.
- **PRIORITY** – *optional* – A numeric value used to decide which resource will be used when multiple resource records compete for a single target. Numerically higher values have higher priorities. Without a priority defined, if two or more resources reference the same target, the last authorized resource will be used.
`PRIORITY=2`
    - **Disk** – Defines the priority of this resource record. For example, you might define a network share that will map to "T:" for most users, and define it with a priority of 1. If a small group of users require a different connection to their "T:" drive, you could define their resource record with a priority of 2 and their resource will take precedence when they log in.
    - **Print** – Used in a similar fashion to Disk resources, but applicable only when an LPT# device name is used as a target.
- **ERRORCONTROL** – *optional* – A parameter that controls how errors that occur while mapping resources are handled. Normally, resource failures are displayed in red and cause the helpdesk message to be displayed at the end of the script, enabling a 15-minute delay. A value of WARN displays the connection and error in yellow. A value of IGNORE displays the connection in green with "NOT AVAILABLE". Either setting suppresses the helpdesk message and error delay. This is recommended for use where the &USER& macro is used and the target resource might not be available for every user.
    - **Disk & Print** – supported.
    - **Message & Command** – not supported as non-existent paths are ignored.

Resource Access-Management Parameters

The next group of parameters permit or reject access to the resource. If none of the parameters are defined, the user will be permitted access. If any of the parameters defined prevent access to the resource, a connection to the resource will not be made, even if other parameters allow it, except for the Users parameter. This follows the general concept of "most restrictive" security. These values are supported by all resource classes. Note that this controls which resources will be connected or used and does not actually secure the resource. Resource access can only be restricted by NTFS permissions.

- **USERS –** *optional* – A string value containing a list of specific user IDs who are authorized to access this resource. When the current user ID is matched to one of the string elements, ***the resource is permitted and all other authorization checks are bypassed***. This allows per-user mapping to be easily defined.
  `USERS=jsmith,mjones`

- **PRIV** – *optional* – A general permission level that controls access to the resource. The value, either ADMIN or NOGUEST, provides a gross level of access control to the resource. When set to ADMIN, access is permitted only when the user has admin-level access (Local or Domain). When set to NOGUEST, access is not permitted to users with Guest level access. This is useful in a workgroup environment where network groups are not available or for very small networks where you want to apply simple restrictions to the shares. This works particularly well with the Message resource type to display a message that reminds anyone with Admin rights to use extra care during their login session.
  `PRIV=Admin`

- **GROUPS** – *optional* – A comma-delimited list of AD Groups whose members are permitted access to the resource. If the user is a member of *any* of the listed groups, a connection to the resource will be permitted. There are modifiers available to this parameter that change the group membership association from *permitted* to *mandatory*. If mandatory membership is enabled, the user must be a member of every specified group. A second modifier performs *exclusion*, which defines that the user must NOT be a member of the specified group(s). There are many combinations of group specifications and modifiers to tailor the access to the resource. These modifiers are discussed in detail later in this manual.
  `GROUPS=Administrators,Marketing,Finance`

- **OUS –** *optional* – A comma-delimited list of OU names. Permits mapping when the user is a member of any of the specified User OUs. Note that this requires a consistent OU structure in AD to be effective.
  `OUS=Sales,Finance`

  See the section "*Group and OU Processing*" section for complete information related to configuring GROUPS and OUS parameters.

- **COMPUTEROUS –** *optional* – Similar to the OUS parameter, it permits the mapping when the user's PC is a member of any of the specified computer OUs. Again, a consistent AD structure is recommended to implement this effectively.
  `COMPUTEROUS=Campus1,Campus2`

- **SITES –** *optional* – A comma-delimited list of Active Directory site names. Members of a site that appears in the list will be permitted access to the resource.
  `SITES=New York,Los Angeles`
- **ADATTR** – *optional* – A list of attribute-name to value-list parameters. This provides the ability to map a resource by associating AD Attributes with specific sets of values. See the "*AD Attribute Processing*" section for full details.
  `ADATTR=+,Department:Sales;Marketing,L:New York;Chicago`
- **CONNTYPE** – *optional* – A comma-delimited list that defines the permitted connection types – CON (Console), RDP (Terminal Server), or ICA (Citrix). The connection type is determined when the login script starts. If the current connection type can be matched with the parameters specified in the list, a connection to the resource will be permitted. If CONNTYPE is undefined, than any method is allowed. *Do not use the global CONNTYPE setting if per-resource controls are desired!*
  `CONNTYPE=ICA`
- **LAPTOP** – *optional* – A Y/N Boolean setting that, when defined, restricts the resource to laptop or non-laptop systems. WMI is used to determine if the system contains an active battery. A "Y" value will restrict the resource to laptop systems, while an "N" value will restrict the resource to non-laptop systems. The default (undefined) is to not restrict the resource.
  `LAPTOP=Y`
- **LOGIC** – *optional* – A string value containing "OR" which is used to control how User Group and User OU access controls are resolved. Under normal conditions, a user must meet the requirements of all qualifiers. In the case where both Group and OU parameters are defined in a single resource record, the user must be qualified by both sets of parameters. There are times where this is too restrictive – the share should be mapped by either OU or Group membership. Setting the LOGIC value to "OR" will relax the matching requirements, requiring just an OU *or* Group parameter match.
  `LOGIC=or`

  Note that using an OR Logic parameter with mandatory group membership will fail if the user is not a member of a specified mandatory group. Defining mandatory groups and an OU list with OR logic – while technically not an error – would have no practical value.
- **CFGSET** – *optional* – A string value containing an arbitrary name, used to group specific resource records together. When a resource record contains a CFGSET value, it is processed only when the command-line parameter `--C` *`name`* is specified and the name matches the CFGSET name in the record. This allows a single config file to contain both common and department or region-specific records without having to use OU or Group controls. Resource records without a CFGSET are always processed, but only records with CFGSET defined that match the command line argument will be evaluated.

Special Resource Parameters

The following parameters are used only by certain resource classes to modify the connection actions or preferences. USER_*name* parameters are not supported by the GUI admin tool.

- **USER_*name*** – *optional, all classes* – User-defined resource record value, used by the UserProcess.kix script. Multiple *name* values are permitted. The example below, based on the example code, targets workstations whose names begin with "ny02b17".
  ```
  USER_WKSTN=ny02b17
  ```
- **MAKELINK** – *optional, Disk class* – A Boolean value that when true, creates a network shortcut in Network Neighborhood in addition to mapping to a drive letter. This value is ignored/forced on when TARGET=UNC.
- **DELAY** – *optional, Message class* – Used to override the default 2-second delay. Any positive value can be specified, including zero for no delay at all.
  ```
  DELAY=5
  ```
- **COLOR –** *optional, Message class* – Used to define the color of the foreground text in message displays. Allowed color definitions are Blue, Green, Cyan, Red, Magenta, Yellow, and White. Only the bright colors are supported. If the color specified is invalid, or is not specified, White will be used. The background color cannot be set.
  ```
  Color=Cyan
  ```
- **PROMPT** – *optional, Message class* – Displays a prompt in a pop-up window and allows a Yes/No action to be taken. Prompt is processed after the message file is displayed, allowing a fairly large amount of text to be displayed during the logon process. After the message is displayed, if Prompt is defined, the message is displayed in a System Modal dialog box (blocks all other processing) and waits for a button to be clicked before proceeding. The Delay value is ignored if Prompt is defined. The Prompt parameter takes 5 parts, delimited with semicolons.

  ```
  PROMPT=type;title;message text;YES_action;NO_action
  ```

o Type	0 (zero) for an "OK" button or 1 (one) for Yes/No buttons.
o Title	A title for the dialog box – 64 chars max recommended
o MessageText	The text to be displayed in the dialog box. Up to 1024 characters are allowed, all on one line. Line breaks can be defined by the sequence "\n" anywhere in the text.
o YES_Action	The action to take if the Yes button is clicked
o No_Action	The action to take if the No button is clicked.

  Actions are supported only for Yes/No type of prompts. No action is available for the simple OK dialog box. If no action is defined, none is taken and processing continues.

  The only action currently supported is "LOGOFF", which will forcibly end the user's login session. This allows an "Acceptable Use" message to be displayed. If the user accepts the policy (responds "Yes"), the login continues. If they respond "No", the login terminates.

- **ARGS –** *optional, Command class* – Any arguments required by the defined command. Environment variables may be referenced in the argument string. Value Rewriting may be used with the ARGS parameter, with one caveat. When Value Rewriting cannot find a matching lookup value, the entire parameter value is cleared (since a value without its lookup component is likely invalid). This means that ALL arguments will be cleared if a lookup cannot be resolved. To prevent errors, the logic employed will then clear the COMMAND entirely if a Value Rewrite lookup parameter cannot be resolved.
  ```
  ARGS=script.kix --r
  ```
- **METHOD** – *optional, Command class* – Defines the method of execution from within KiXtart.
  ```
  METHOD=Run
  ```
  - **Call** – calls another KiXtart script in the current script context (variables are preserved in the new KiX script!)
  - **Run** – runs an external process, the KiXtart login script does not wait for the specified command to complete.
  - **Shell –** runs an external process, the KiXtart login script waits for the specified command to complete. This is the *Default Method!*
- **RUNONCE –** *optional, Command class* – Causes a command to be run only one time, per user, per computer when set to true. The Resource Name is used as an identifier. This identifier is used to create a key in the HKCU registry path with the current date to indicate that the command has been performed. If the value exists and contains data, the command will not be run on future logins. The date is not checked.
- **SETDEFAULT** – *optional, Printer class* - Defines the resource as the default printer. The printer name and share name must be the same for this option to work. This value supports the following parameters:

Y	The printer will be marked “Default Eligible”
N	(or undefined) The printer will not become “Default Eligible”
Soft	The printer will become “Default Eligible” if a default printer is not already defined
ByGroup:list	The printer will become “Default Eligible” if the user is a member of a group defined in the list.
ByOU:list	The printer will become “Default Eligible” if the user is a member of an OU defined in the list.
ByCOU:list	The printer will become “Default Eligible” if the computer is a member of an OU defined in the list.
ByCGroup:list	The printer will become “Default Eligible” if the computer is a member of a group defined in the list.

  In all cases, “list” is a comma-delimited list of AD objects. Unlike the OUS, COMPUTEROUS, or GROUPS parameters, the “+” and “!” qualifiers are not permitted and, if used, will cause the match to fail.

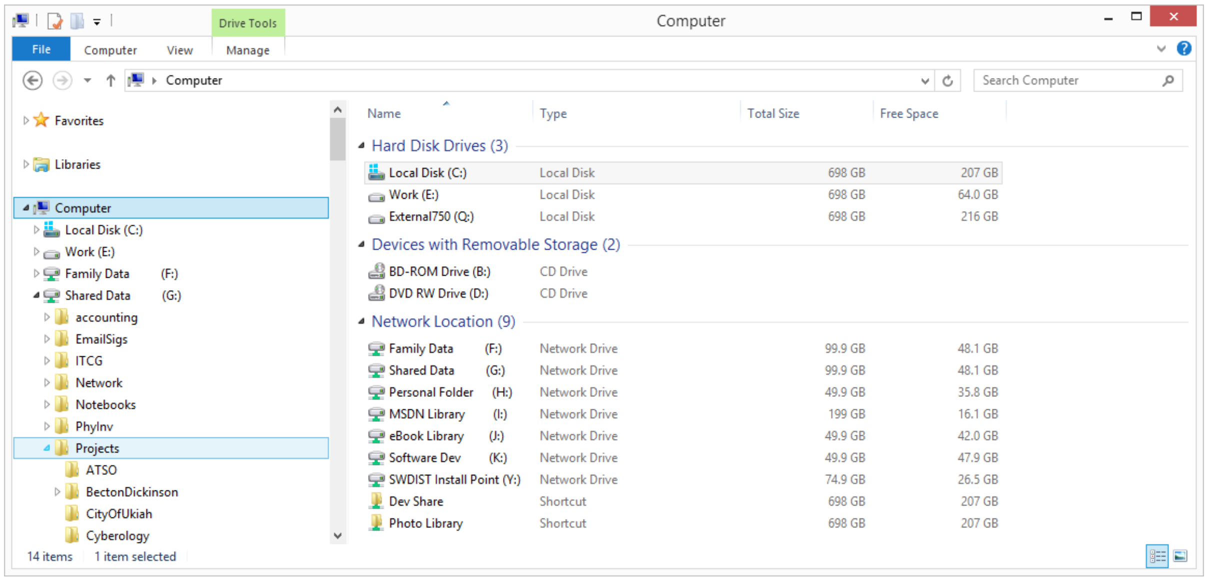
```
SETDEFAULT=ByOU:Brooklyn Users,Westchester Users
```

Network Shortcut vs Drive Mapping

In addition to traditional drive mapping, where a remote resource is mapped to a local drive letter, the script can pre-populate the Network Neighborhood with shortcuts to remote resources.

One of the key advantages to this is that there are no disk drive letter limitations – any number of shortcuts can be processed. Mapped drive letters can be augmented with shortcuts for sites desiring to migrate away from mapped drives by enabling the MAKELINK parameter in each drive resource record. Note that because the drive letter and path for a HOME reference is controlled by Active Directory, the MAKELINK parameter is ignored if the PATH=HOME.

The image below illustrates the mapped drives and network shortcuts in the Network Locations section of Explorer.



- Favorites
- Libraries
- Computer
  - Local Disk (C:)
  - Work (E:)
  - Family Data (F:)
  - Shared Data (G:)
    - accounting
    - EmailSigs
    - ITCG
    - Network
    - Notebooks
    - PhyInv
    - Projects
      - ATSO
      - BectonDickinson
      - CityOfUkiah
      - Cyberology

Name	Type	Total Size	Free Space
Hard Disk Drives (3)			
Local Disk (C:)	Local Disk	698 GB	207 GB
Work (E:)	Local Disk	698 GB	64.0 GB
External750 (Q:)	Local Disk	698 GB	216 GB
Devices with Removable Storage (2)			
BD-ROM Drive (B:)	CD Drive		
DVD RW Drive (D:)	CD Drive		
Network Location (9)			
Family Data (F:)	Network Drive	99.9 GB	48.1 GB
Shared Data (G:)	Network Drive	99.9 GB	48.1 GB
Personal Folder (H:)	Network Drive	49.9 GB	35.8 GB
MSDN Library (I:)	Network Drive	199 GB	16.1 GB
eBook Library (J:)	Network Drive	49.9 GB	42.0 GB
Software Dev (K:)	Network Drive	49.9 GB	47.9 GB
SWDIST Install Point (Y:)	Network Drive	74.9 GB	26.5 GB
Dev Share	Shortcut	698 GB	207 GB
Photo Library	Shortcut	698 GB	207 GB

14 items 1 item selected

Group and OU Processing

Using Active Directory Groups and OUs to perform resource processing can provide a high degree of configuration flexibility. All four classes of resource records support Group and OU processing. There are four forms of processing, as described below. The following examples show group based mapping, but apply the same way to OU based mapping. Group and OU processing can be used separately or together, permitting complex configurations to be achieved. In addition, Computer OU membership can be used independent of User Group & OU matching.

1. The simplest form – Allowed Groups or OUs
   The form `GROUP=group1,group2,group3…` or `OUS=ou1,ou2…` will cause the resource record to be permitted if the current user is a member of at least one of the listed groups or OU names. The resource record will be ignored if the user is not a member of any group or OU listed. Note that an “OU” is a simple name, such as “Chicago”, which might be part of the DN `OU=users,OU=Chicago,OU=America, DC=contoso,DC=com.` The OU name is qualified internally be prefixing it with “OU=” and adding a comma suffix.

2. Mandatory Groups
   A modifier can be used to require membership in two or more groups.
   The form `GROUP=+,group1,group2` would require that the current user be a member of *both* groups. The leading “+” is used to indicate mandatory membership in all listed groups, and can be used alone or as the leading character of any group name. The “+” modifier has no value if only one group name is defined. A special form of Mandatory Groups combines Mandatory and Allowed forms. The form `GROUP=+,group1,(group2;group3)` defines a mandatory group and a mandatory list – the resource is mapped if the user is a member of group1 AND any of the groups in the list in parenthesis. Note that the sub-group is delimited with semicolons.

   Since a user can only be a member of one OU, this attribute does not apply to OU processing.

3. Membership Negation (Exclusion)
   A “!” modifier can be used to indicate negation. If a “!” is placed in front of the argument name, the resource would be processed if the user was *not* a member of the named group or OU. For example, `GROUP=!group1` would process the resource for everyone except members of group1, while `GROUP=+,group1,!group2`
   would process the resource for anyone who was a member of group1 but NOT a member of group2. Note that for OU processing, being a member of an OU that is excluded will result in access to the resource being denied.

4. Action Negation (Reversal)
   If a “!” modifier is placed at the end of the argument name, it reverses the action of the resource. For example: `GROUP=group1!,group2`
   would prevent the resource from being processed if the user was a member of group1, even if they were a member of group2. Action Negation is most useful when used without the Mandatory flag.

Active Directory Attribute Processing

Active Directory Attribute processing provides an effective way to authorize resources based on standard AD Attributes, such as Department, City, or even Title. The ADATTR parameter can be used with all four classes of resource record.

The general format of this parameter is

```
ADATTR=ATTR_NAME:Value_List[,ATTR_NAME:Value_List]
```

ATTR_NAME is any standard AD/LDAP attribute – see the section "Common AD Attributes" for a list of the most common attribute names. The value of this parameter is extracted from the user’s AD object and compared to the Value_List string.

VALUE_LIST is a semicolon-delimited list of values that are allowed. If the value of the attribute associated with the current user is found in the Value_List string, the resource is permitted.

Multiple Name:Value_List sets can be defined, separated with commas. The ADATTR parameter accepts the same modifiers as the Group parameter to allow mandatory association with all Name:Value_List sets, Membership Negation, and Action Negation. Note that there is one distinct difference between Group modifiers and AD Attribute modifiers – each Name:Value_List set can accept a modifier, just like a group parameter. The Value_List is always a “match any”, since a user has only one value associated with an attribute. That is, the form:

```
ADATTR=+,Department:Sales;Marketing,L:NYC,Chicago
```

invokes a “mandatory” flag requiring both a Department and City (L) match, but Department can be either Sales or Marketing, and City can be either NYC or Chicago.

Combined Logic of Group, OU, and AD Attribute Parameters

The greatest power of Group, OU, and Attribute Processing comes when you combine the logic. You can process a resource when a user is a member of one group AND expressly not a member of another group (GROUP=+,group1,!group2). Many complex variations are possible. The sample configuration file shows how most users map a data share to G:, but certain users map another share to G: and the data share to F: using group-based processing.

You can extend this by specifying more than one set of parameters. For example, a resource can be available only when a user is a member of a specific group and a particular OU, or is in an OU and has a specific title.

The default logic when more than one of the Group, OU, and Attribute parameters are specified is “AND”, requiring that all of the parameters to be validated. Setting the Logic parameter to “OR” will allow the resource to be processed if any of the Group, OU, or Attribute parameter requirements have been satisfied. If any of the Group, OU, or AD Attribute parameters are not specified, they do not play a part on the authorization logic. Thus, Groups, OUs, and AD Attributes can be used singly or in any combination to achieve the desired result.

Connection Processing – CAUTION!

Each resource record has several parameters that control the processing and ultimate authorization of the resource for mapping. These include PRIV, GROUPS, OUS, ATATTR, SITES, and CONNTYPE. All of these are optional, but if any are specified, the conditions must be met. Multiple parameters can be defined, in which case, *all* conditions must be met. It’s possible, therefore, to specify that a resource requires Admin privilege, membership in a specific group or groups, membership in a specific OU, and only when logging into a Citrix connection. Clearly, care must be exercised when specifying more than one parameter set!

When things don’t work as expected – simplify! It’s best to approach defining resource record access properties in a minimalistic fashion. Use just an OU list, or a group list when possible. While there are enough options to work out nearly any access scenario, there are enough options to get you into trouble, too.

Note that the default method to process a record containing both Group and OU definitions is to permit access only if a user is a member of the defined group(s) *and* a member of one of the defined OU(s). The optional parameter LOGIC=OR can modify this method. In the case of LOGIC=OR, the user can be a member of one of the defined OUs *or* one of the defined groups. This can reduce the number of resource records for shares that must be mapped based on either OU or Group membership. The use of mandatory groups should be avoided when employing OR based logic.

When you employ Computer OU matching, the logic is independent of User group or OU matching. That is, it is equivalent to specifying a Site, Priv, or other qualifier. If you specify a Computer OU and no match is found, the resource is not mapped, regardless of whether User OU or Group matching would allow a connection.

Carefully consider the use of mandatory, negation, and inversion operators, and how they interact with the LOGIC setting. Most important is the understanding of the exclusion operator (!object).

- When evaluating groups, the Mandatory flag decides if exclusion denies access to the resource or not. If Mandatory is not set, and the user is a member of an excluded group, the script will *ignore* the setting and continue looking for other group matches. If none are found, the resource will be skipped, but if a match is found in another group in the list, the resource will be mapped. When Mandatory is set, membership in an excluded group will immediately deny access to the resource.
- When evaluating User or Computer OUs, the Mandatory flag has no effect, since an object may not be a member of multiple OUs. Special processing is done when multiple OUs are listed with exclusions which has the effect of the Mandatory flag being set. That is – if the object is a member of *any* excluded OU, the access is denied and additional processing is not performed.

Basically, if you use excluded groups, processing is relaxed unless you use the Mandatory flag. If you use excluded OUs, the processing is strictly enforced.

Value Rewriting & Lookup Records

Value Rewriting is a powerful capability that allows the administrator to dynamically adjust the content of a resource parameter based on a user, group, AD-Site, OU, or even network subnet. Two common uses for this are mapping each department's share to the same drive letter, and mapping printers based on subnet or group. Value Rewriting technology is available to PATH, DESC, ARGS, and LANGUAGEID parameters.

Lookup records are used by the Value Rewriting feature. When the parameter contains a special Value Rewrite Lookup token, the value is changed to reflect the values in the corresponding lookup record.

For example, the PATH value defined by the resource records usually references a specific location within the network. There are times, however, when a resource needs to be tailored to a specific user, AD Site, User OU, Computer OU, or Network Subnet. Value Rewriting provides several different methods for altering the resource path.

Path Rewrites are triggered by a token in the parameter value with the format

```
&TYPE[:lookup]&
```

"TYPE" represents one of the rewrite types listed below, and "[:lookup]" is an optional parameter that defines the lookup table to use. Most rewrite types allow both Direct and Lookup substitutions, the exception is Subnet, which only allows Lookup substitutions.

Direct substitutions are basic rewrites that simply replace the macro text with the name associated with the TYPE. Lookups are table-based substitutions that replace the macro with the value determined by a lookup table.

Lookup-based rewrites are the most powerful method, and an example might best illustrate the capability. This example of USER based lookup rewrites uses the UserID to locate a value in the lookup table.

There are times that JSmith and her team need access to one subfolder of a share, and other users need access to different subfolders. Even though they are a team, there might not be a related AD group that can be used for association with a share. User ID Rewriting combined with lookup records provides this powerful functionality.

A User ID Rewrite record looks like this:

```
PATH=\\server\share\&USER:MAP03&
```

A lookup record, placed anywhere in the configuration file, might contain the following value definitions:

```
[MAP03]
JSmith=project_X
PWhite=project_X
MBlack=project_X
TGreen=project_X
```

Now, when any of those users log in, they will receive a connection to the share "\\server\share\project_X". If an ID Mapping does not exist, the connection to the share is not made. Thus, only the four people listed will connect to that share.

Note that the name defined in the &USER:xxx& must match the name of the lookup table – in the example above, the User lookup table name is “MAP03”. Any name can be used so long as it is unique.

Value Rewrite Types

1. **User ID Substitution**
   Token syntax: `&USER&`

   This method allows the user ID of the person logging in to be placed into the resource path. A lookup record is not used.

   If **`PATH=\\server\share\&USER&`** is defined in the configuration file, and JSmith logs in, the path that will be mapped will be “\\server\share\jsmith”.

2. **User ID Lookup**
   Token syntax: `&USER:table_name&`

   The user’s login ID is used as a lookup value in the defined lookup table. If a value is found, it replaces the entire token. If a value is not found, no connection is made to the resource.

3. **Site ID Substitution**
   Token syntax: `&SITE&`

   This form causes the Active Directory Site name to be substituted for the token. A lookup record is not used.

4. **Site ID Lookup**
   Token syntax: `&SITE:table_name&`

   This form causes the current Active Directory Site name to be used as a lookup value into the named lookup table. If a value is found, it replaces the entire token. If a value is not found, no connection is made to the resource.

5. **User OU Substitution**
   Token syntax: `&OU&`

   This form causes the user’s OU name to replace the token in the string. A lookup record is not used.

6. **User OU Lookup**
   Token syntax: `&OU:table_name&`

   This form causes the user’s OU to be used as a lookup value into the named lookup table. If a value is found, it replaces the entire token. If a value is not found, no connection is made to the resource.

7. **Computer OU Substitution**
   Token syntax: `&COU&`

   This form causes the local computer’s OU name to replace the token in the string. A lookup record is not used.

8. **Computer OU Lookup**
   Token syntax: `&COU:table_name&`

   This form causes the user's OU to be used as a lookup value into the named lookup table. If a value is found, it replaces the entire token. If a value is not found, no connection is made to the resource.

9. **Computer Name Substitution**
   Token syntax: `&HOST&`

   This form causes the computer's name to replace the token in the string. A lookup table is not used.

10. **Computer Name Lookup**
    Token syntax: `&HOST:table_name&`

    This form will cause the computer's name to be used as a lookup value into the named lookup table. If a value is found, it replaces the entire token. If a value is not found, no connection is made to the resource.

11. **Active Directory Attribute Substitution**
    Token Syntax: `&AD:attribute_name&`

    This form queries AD for the *attribute_name* value that is associated with the current user. If a value is found, it replaces the entire token. If a value is not found, no connection is made to the resource. For example, if the user is located in Santa Fe, the token syntax of "`&AD:L&`" will use the attribute "L" (location) to obtain the city name, replacing the token with "Santa Fe". If the AD Attribute is not defined for the current user, the resource will be ignored.

12. **Active Directory Attribute Lookup**
    Token Syntax: `&AD:attribute_name:table_name&`

    This form queries AD for the *attribute_name* value that is associated with the current user. If a value is found, the defined table_name will be searched for a match. If a match is found, the value returned from the table will replace the entire token string. If the AD attribute is null or not located in the lookup table, the resource is ignored.

13. **Network Subnet Lookup**
    Token syntax: `&SUBNET:table_name&`

    This form will cause the local computer's IP address to be logically ANDed with the corresponding subnet mask. The resulting Network Address will be used as a lookup value into the named lookup record. Direct substitutions are not available for subnet based rewrites. All locally defined IP addresses are evaluated. If an exact network match is not found, a default resource can be defined via "0.0.0.0=text". If this default parameter is not specified, the resource is ignored.

Lookup Value Recursion

Value rewrites are processed recursively, allowing one lookup to reference another lookup. Up to ten passes are performed, although more than ten replacements are possible due to how the replacements are processed.

Each parameter that supports Value Rewriting is passed through the MVLookup function. This makes one pass, checking for each of the thirteen replacement values in the sequence described above. Thus, if an OU:map lookup replaces the value with a SUBNET:map lookup, the subnet lookup will be processed in the same pass, as it comes later in the sequence. If that subnet lookup replaces the string with a USER:map lookup, a seconds pass through the MVLookup function will be performed. This will be repeated until 10 passes through MVLookup are performed, or no more lookup values are detected in the string. If lookup values still exist after 10 passes, the resource is ignored.

The need for recursive Value Rewriting is rare, but supported for large, complex environments. Due to the complexity of creating nested Value Rewrite tables, extensive testing is strongly recommended.

When multiple replacements are made to a value, the debug log will show each individual replacement. For example, an OU to Subnet to User nested lookup will be shown in the log as:

```
Processing TEST
    OK to process this resource!
   MVLookup: &OU:test1&\Share
   MVLookup: &SUBNET:TEST2&\Share
   MVLookup: &USER:test3&\Share
 Translated: \\personalFS\Share
 Translated: ByOUbySubnetByUser
```

Note that “MVLookup” is reported 3 times, followed by the “Translated” data that ultimately resulted from the 3 replacements. The second “Translated” data is the DESC value. Since it has no MVLookup values before it, no translations were performed on this data.

Special Consideration for OU-Based Rewriting

When specifying an OU, you *must* define the OUOffset value in the login.ini file. This specifies which OU field in the DN string should be used. For example, if a User’s OU string returns “OU=Users,OU=Support,OU=IT,DC=domain,DC=dom”, the user’s OU will be regarded as “Support”, with an offset of 1. This is a fairly common configuration.

If the structure places the Users at a higher level, such as “OU=Support,OU=IT, OU=Users,DC=domain,DC=dom”, the OU name is still “Support”, but the index is now 0 (zero) because the most specific OU is in the first position.

Most OU structures will use an offset of either 0 or 1. If one value doesn’t work the way you expect, try the alternate. By viewing a standard DN string, ignore the CN field, and count the OU fields (starting with 0) until you find the offset of the OU component you want to use.

The offset value is used only by the Value Rewrite functions because the primary OU that the user is a member of must be determined. When using OUs to determine if a resource should be connected, the OUOffset is not used. Each OU defined in the list is checked against the user's entire DN string to find a match. Thus, if a resource record contained an "OUS=Users" directive, the resource would likely map for every user, since nearly every user has an "OU=Users" component in their DN string!

Using User or Computer OU for authorization and rewriting assumes a well-defined AD structure with all User or Computer objects under a common root and a consistent path depth. Note that the User and Computer DN strings are written to the *LoginDebug.log* file for your reference in choosing an OU Offset value.

Rewriting for DESC values

The same rules for Path Rewriting apply to Description values, including lookups. This is useful when customizing the mapped drive description. For example, if you use OU based Path Rewriting to lookup a department's share, you could set the description to use a simple OU rewrite so the department name is in the description.

```
[Sample Dept Share]
CLASS=DISK
TARGET=G:
PATH=&OU:DEPT&
DESC=&OU& Share
```

```
[DEPT]
IT=\\fileserver1\ITshare
HR=\\fileserver2\HRshare
```

Compound Rewrites

Compound rewrites can be used. For example, you might want to lookup a share name by OU and then specify the subfolder of that share by userid:

```
PATH=&OU:DEPT&\&USER&$
```

Compound Rewrites can also be done through the lookup value itself. For example, in the config file we use for testing, we find:

```
# TEST OF NESTED LOOKUPS - USE WITH CARE!!
# The OU lookup returns a SUBNET lookup, which returns a USER lookup
# the nesting can end at any time by returning a server name
# (see the Region6 and 10.32.80.0 lookup values)
[TEST18]
CLASS=DISK
TARGET=T:
PATH=&OU:test18a&\share
DESC=Map by OU, then Subnet, then User lookups

[TEST18a]
Corporate=&SUBNET:test18b&
Region6=\\fileserver

[TEST18b]
10.32.80.0=\\servefiles
172.16.12.0=&USER:test18c&

[TEST18c]
George=\\fileserver1\Accounting
Mary=\\fileserver1\IT
```

Of course, you should use valid name, subnet, and OU values for testing.

Special Consideration for Subnet Rewriting

When using Subnet Rewriting, you must carefully identify the network address. For example, if a computer’s address is 172.16.1.48 and the subnet mask is 255.255.254.0, the network address is 172.16.0.0. Thus, a lookup record for this network might look something like 172.16.0.0=\\server0\share.

0.0.0.0=\\server\share can be used as a default replacement value, as this will match with any network that does not have a more specific peer.

Note that you cannot specify subnets or supernets, as the network address is calculated based on the local computer’s address and netmask. If multiple subnets need access to the same share, you will need to create multiple lookup records, one for each physical subnet.

Custom Resource Processing

There are certain situations where even the versatility of the standard script cannot meet all of the specific requirements to control resource access. User-defined processing is a method that allows the administrator to write their own code and have it executed by the login script. Two sample scripts have been provided that perform similar workstation name matching. One is very basic and the other is a bit more complex.

Resource Processing Concepts

The login script reads each resource record and utilizes the following sequence to decide if access to a resource should be granted. As soon as a deny access condition is met, no further checks are performed and the next resource record is evaluated. The process repeats until all resource records have been examined. The process for evaluating a single resource is:

1. Assume that access is permitted.
2. If USERS is defined and matched, permit the resource and ignore remaining tests.
3. If CFGSET is defined and does not match the parameter, ignore the resource.
4. If PRIV is defined and not met, ignore the resource.
5. If CONNTYPE is defined and not met, ignore the resource.
6. If LAPTOP is defined and condition is not met, ignore the resource.
7. If SITES is defined, check site membership and ignore the resource if not a site member.
8. If COMPUTEROUs is defined, perform Computer OU matching. ignore the resource if no match is found.
9. If GROUPS is defined, evaluate membership. Set GROUP allow-access-flag if requirements are met; continue processing.
10. If OUs is defined, check membership, Set OU allow-access-flag if user is a member; continue processing.
11. If AD Attributes parameter is defined, check for a match, Set ADA allow-access-flag if a match is made; continue processing.
12. Evaluate LOGIC value
    a. If LOGIC is “AND” or undefined and any of the GROUPS, OUS, or ADA allow-access-flags are not set, ignore the resource.
    b. If LOGIC is “OR” and none of the GROUPS, OUS, or ADA allow-access-flags are set, ignore the resource.

Note that checks 9 through 11 do not actually set a Deny until step 12. At this point, if no means to deny access to the resource has been determined, the resource can be processed.

13. Perform optional User-Defined process
    a. If the $USERPROCESSACTIVE value is true, call the UserProcess() function, passing resource record ID and Class values.
    b. The user defined function runs, returning 0 if resource is permitted or 1 if it is denied.

14. If resource access is allowed, add the resource to the allowed list and continue processing additional records.

Configuring User-Defined Resource Processing

There are three requirements to implement user-defined processing:

1. The Kixtart script *UserProcess.Kix* must be present in the login script folder where Kix32 and the kixtart.kix script reside.
2. The script must set a global variable “$USERPROCESSACTIVE” to a non-zero value (ie: 1). The variable must be set as soon as the script is called and not by the function so that it can indicate that user-defined processing should be used.
3. The script must contain a function called UserProcess, which must accept two arguments – the name of the resource record and the resource class value. It must return a value of 0 ($UserProcess=0 and not “Exit 0”) if access to the resource is permitted or a value of 1 if access is denied.

A sample UserProcess.kix script is provided which illustrates several key methods, including:

- Properly defining the USERPROCESSACTIVE variable value.
- Accessing the current resource record values.
- Use of USER_*name* values in the resource record.
- Comparing part of the workstation name to the user-defined value.
- Returning values to allow or deny access.

The actual code implemented by the user can be as simple or complex as required, so long as the three key requirements are met. While a single method is shown in the example script, multiple methods can be implemented by using a USER_ACTION value. This can then define which block of code or functions should be executed on a record by record basis.

The *UserProcess.kix* script may contain additional user-defined functions to support the requirements. Care should be exercised to avoid duplicate function and global variable names! Review the *readme.txt* file for a list of function and global variable names. Of course, the user is free to call any internal function they may find useful, and *read* any global variable. Changing global variables may cause unpredictable results and should be avoided. The login script contains many public functions that could be useful in your custom logic.

Using the UserProcess.Kix File

User-defined processing is performed after all standard processing has been completed. To rely only on user-defined controls, no other qualifiers should be defined in the resource record.

The record name and type are passed to the UserProcess function. This allows the function to quickly determine the resource type (disk, print, command, or message) and read additional parameter values from the resource record. Any number of additional parameters may be defined, although each should begin with "USER_" to avoid any current or future conflicts.

The best way to understand user-defined processing is by a simple example. In this example, the first 8 characters of the computer name are examined. These are compared to a list that is read from the resource record. The example record is shown here:

```
[Lab207InstructorDisk]
CLASS=DISK
TARGET=I:
PATH=\\server\sharename
DESC=Instructor - Lab 207
USER_HOSTQUALIFIER=COEH207I
```

The "HostQualifier" format is SSBBRRRT##, where SS represents the site (Covington campus), BB the building (Edwards Hall), RRR the room (107), and T the type (I for instructor, S for student). The example script checks the first 8 characters of the local workstation name against the HOSTQUALIFIER list. Any computer NOT beginning with those 8 characters would be denied access. Since only the instructor's PC would have the "I" in position 8, a resource could be mapped only for that PC in that specific room. In a larger classroom, there might be several instructor PCs, so any/all would map the resource.

If classrooms 206 and 207 both needed the same resources on the instructor machines, the HostQualifier string would contain "COEH206I,COEH207I", and either would then match the first 8 characters of the workstation name.

The example is fairly simple, but any level or complexity of logic may be employed. The general concept is to determine if the user or computer should be denied access to the resource. If so, simply return a True value.

Customizing Language-Specific Messages

Language files allow the messages to be customized on a user by user basis. Up to 190 distinct languages can be supported, based on standard language locale values. English, German, Spanish, Polish, Dutch, and Swedish languages are currently built-in to the core script using the standard message format.

To define a new language, copy the supplied *_UserLang.kix* sample file to a new filename using the format *LSL_####.LNG*. Be sure to replace the "####" with the language locale ID. Locale IDs are listed in Appendix 1 for your reference. Edit the file, replacing the sample English messages with the appropriate language-specific text. Be sure to retain the macros (&1&) in the message strings, as these are replaced with data from the PC, user environment, or network environment. Pay close attention to the structure of the variable definitions, quoting the strings with single quotes and placing the macros in the correct location within each string.

Despite having a .LNG file extension, these are indeed Kixtart scripts, and can be tokenized to prevent anyone from modifying them without authorization. In fact, tokenizing the file after testing is complete is highly recommended. Be sure to tokenize the file with the same version of Kix32 that you will run the script with.

The language files should be placed into the same shared folder as the rest of the login script files. If a language file cannot be located when needed, English messages will be displayed.

Sample .LNG Message File

The comments at the top of the script below illustrate the intended message and the values that will replace the macro tags in the actual message strings. When you modify the message file, be sure to edit the $MSGTXT variables and not just the comments!

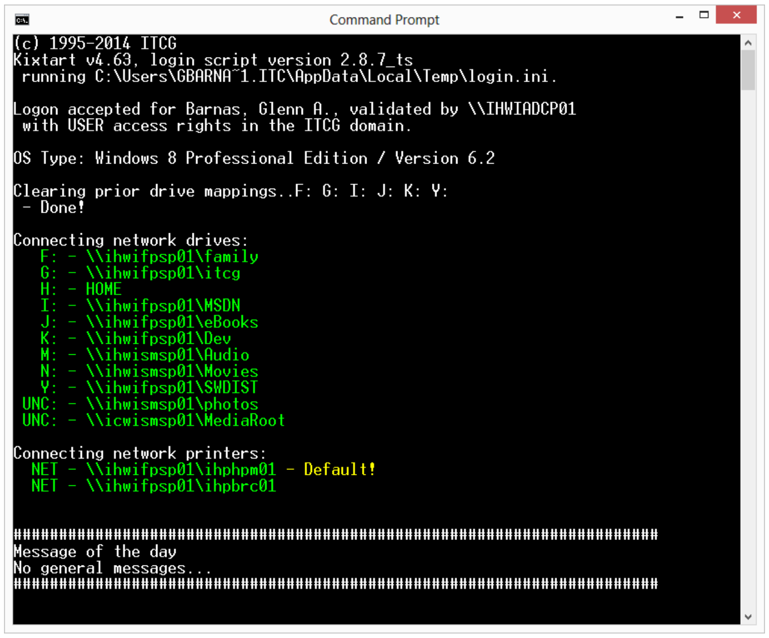
Pay particular attention to the use of apostrophes and other grammatical marks when creating a new language file!

```
; 'Kixtart v' @KIX ' login script processor version ' $VERSION
; ' running ' $INIFILE '.'
;  'Logon accepted for ' @FULLNAME ', validated by '@LSERVER
; ' with ' @PRIV ' access rights in the ' @DOMAIN ' domain.'
; 'OS Type: ' @PRODUCTTYPE ' / Version '  @DOS
;
$MSGTXT[0] = 'Kixtart v&1&, login script version &2&'
$MSGTXT[1] = ' running &1&.'
$MSGTXT[2] = 'Logon accepted for &1&, validated by &2&'
$MSGTXT[3] = ' with &1& access rights in the &2& domain.'
$MSGTXT[4] = 'OS Type: &1& / Version &2&'
;
; Drive, Printer, and Command messages
$MSGTXT[5] = 'Clearing prior drive mappings.'
$MSGTXT[6] = 'Clearing prior printer mappings.'
$MSGTXT[7] = ' - Done!'
$MSGTXT[8] = 'Connecting network drives: '
$MSGTXT[9] = 'Connecting network printers: '
$MSGTXT[10] = ' - Default!'
$MSGTXT[11] = 'Please wait while &1& runs...'
$MSGTXT[12] = 'Errors encountered during login processing!'
$MSGTXT[13] = 'Invalid domain, script terminated.'
$MSGTXT[14] = 'Waiting &1& seconds for NAC to grant access.'
$MSGTXT[15] = 'Timeout while waiting for NAC to grant access, script terminated.'
```

The $MSGTXT global array is declared to hold 30 elements (0-29). Elements 5-15 are used by different parts of the script and their purpose and format should not be changed. If you will be using the *UserDisplay.kix* script to customize the login message screen, you can redefine the messages held in elements 0-4 and freely use elements 20-29 for your own purposes. Elements 16-19 are reserved for future use.

The Standard Display Screen

The default login screen will look something like this:



```
(c) 1995-2014 ITCG
Kixtart v4.63, login script version 2.8.7_ts
 running C:\Users\GBARNA~1.ITC\AppData\Local\Temp\login.ini.

Logon accepted for Barnas, Glenn A., validated by \\IHWIADCP01
 with USER access rights in the ITCG domain.

OS Type: Windows 8 Professional Edition / Version 6.2

Clearing prior drive mappings..F: G: I: J: K: Y:
 - Done!

Connecting network drives:
   F: - \\ihwifpsp01\family
   G: - \\ihwifpsp01\itcg
   H: - HOME
   I: - \\ihwifpsp01\MSDN
   J: - \\ihwifpsp01\eBooks
   K: - \\ihwifpsp01\Dev
   M: - \\ihwismsp01\Audio
   N: - \\ihwismsp01\Movies
   Y: - \\ihwifpsp01\SWDIST
 UNC: - \\ihwismsp01\photos
 UNC: - \\icwismsp01\MediaRoot

Connecting network printers:
  NET - \\ihwifpsp01\ihphpm01 - Default!
  NET - \\ihwifpsp01\ihpbrc01


#######################################################################
Message of the day
No general messages...
#######################################################################
```

There are several important pieces of information displayed here. On the second line, we see the version of Kixtart (4.63) used to run the login process, along with the version of the login script (2.8.7). The "_ts" on the end of the script version indicates that timestamps are enabled and performance metrics are being written to the LoginPerf.log file. The path to the config file used is then displayed. This will point to the locally cached copy of the config file.

A standard set of identification text is displayed next. The user's full name is displayed, along with the domain controller that authenticated the user, their access rights, and the domain they logged in to. Operating system version information is displayed on the next line.

If the option to clear drive mappings is enabled, the drive connections that were removed will be listed. Network Shortcuts will be removed silently if this feature is enabled. The drive mappings are then displayed, with drive letter and mapped path. Network shortcuts are created silently when defined to duplicate a drive mapping. Standalone Network Shortcuts are listed on the status screen with "UNC" instead of a drive letter. The HOME mapping (H:) normally will not show the path when it is mapped during the login process. If drive mapping fails, the resource is displayed in red with an error message. Network printers are then mapped and the default printer identified.

Any messages that are configured are then displayed with a band of "#" characters at the start and end, and a band of "=" characters in between each message. Finally, any commands defined will be run. You may see the "Please wait…" message along with the output of any command sent to the screen.

Customizing the Initial Display Screen

The login script can be configured to display as much or as little general information as possible. While most users tend to ignore this (they usually know who they are!), the information is very useful to the help desk and technical staff.

When the login script first starts, it displays some information about the Kixtart executable and script versions, INI file location, User and privilege info, Domain name, and O/S info. This information is fairly basic. Before displaying this information, the screen is cleared and text color set to bright white on a black background. The result of the resource connections are displayed in green if they are successful, and red if errors occur.

The display can be easily customized by placing a *UserDisplay.Kix* script in the same folder as the login script files. If this file is present, it will be used instead of the default messages. A sample file is included in the login script package (which is identical in content and format except for the color, so you can tell that the external script is being used). Note that if a custom message script is used, you are responsible for the coding necessary to display any alternate languages – the built-in language support will not work.

The script should be limited to output statements that display informational messages in specific colors. The use of complex code at this point is discouraged as it may interfere with or be affected by script initialization. Use of the AT command to position the cursor is also discouraged, as further messages indicating the status of mapped drives and printers, as well as the output of message files and possibly even commands will follow.

Acceptable commands include:

- Color
- All Kixtart macros (Be aware that NoMacrosInStrings is active!)
- Output strings and the newline command (?) or macro (@CRLF)
- Defined variables – see the readme.txt file for a list of variables of interest. (Note that NoVarsInStrings is active!)
- ReadProfileString($INIFILE, 'CUSTOM', valuename)
  You can add values to the CUSTOM section of the LOGIN.INI file, and this will output them to the screen. If you read the parameter into a variable, you *must* declare your variable before it is used. (Explicit is active – declare your variables!)
- MessageBox()
  This might be a good place to place an acceptable-use message and require acceptance by the user before logging in. Be sure to use a timeout, or avoid using this login script feature with any account running automated tasks.
- Beep
- GetDiskSpace – to generate a warning about low disk space on the local drive(s).

Note that the use of other commands at this point could affect the operation of the login script. If you have any login issues after implementing this script, we strongly recommend that you rename this file and use the default messages to determine if the display script is causing the problem.

Performance Logging

The performance logging capability of the login script can be leveraged within your UserProcess function. The TimeStamp(*ID*) function writes a time-stamped entry to the log with the “ID” string that was provided as an argument. A call should be made upon entry to your function and upon start of any key process, and again upon exit. This can help narrow down any performance bottlenecks. Remember that performance logging is only active when the LSPERF.TXT file is present in the startup folder. Note that the resolution of the time is +/- 8ms (0.008 seconds). Refer to the Performance Analysis section of this manual for more information on enabling performance logging.

Debugging

Two levels of debugging are provided to assist in development of your configuration or diagnosing user login problems.

Level 1 Debugging

With level 1 debugging enabled, the login script runs normally, but writes diagnostic information to %USERPROFILE%\LoginDebug.log. This level of debugging is actually enabled simply by creating the %USERPROFILE%\LoginDebug.log file on the target system. The data in this log is overwritten each time a user logs in to prevent the file from growing continuously.

Note the OU String, OU Offset, and User OU values, which can be used to verify that the OU based Path Rewriting is configured properly.

A diagnostic log generated from the sample *login.ini* is shown below (some data obscured):

```
Updating cache...
UserProcess.kix has been loaded and is active.
NAC Checking is ENABLED.
=========================================================
2014/10/11  -  11:54:18
          Kixtart: 4.63
          Version: 2.8.7_ts
         INI File: C:\Users\GBARNA~1.ITC\AppData\Local\Temp\login.ini (updated)
             User: Barnas, Glenn A. / gbarnas
Locale / Language: 1033 / English (United States)
      LoginServer: \\IHWIADCP01
           Domain: ITCG
       User OU DN: OU=Employees,OU=User Accounts -
Regular,OU=Users,OU=Headquarters,DC=itcg,DC=pvt
   User OU Offset: 1
          User OU: User Accounts - Regular
  Wkstn OU Offset: 0
      Wkstn OU DN:
OU=Desktop,OU=Workstations,OU=Computers,OU=Headquarters,DC=itcg,DC=pvt
         Wkstn OU: Desktop
       Logon Mode: 0
           Laptop: 0
     Detected O/S: Windows 8 Professional Edition / Version 6.2
        Privelege: USER
  Local Privelege: User
        ScriptDir: \\itcg\netlogon / \\itcg\netlogon
         StartDir: \\itcg\netlogon
     User Process: 1
     Session Type: RDP-Tcp#0
      LP Hard Map: 0
      Debug Level: 1

 Group Membership:
                    ITCG\Domain Users
                    Everyone
                    IHWD001\Users
                    IHWD001\Remote Desktop Users
                    INTERACTIVE
                    CONSOLE LOGON
                    Authenticated Users
                    This Organization
                    LOCAL
                    ITCG\OG-Development Team
                    ITCG\OG-Web Admins
                    ITCG\OG-Principals
                    ITCG\AC-RD Access - All
                    ITCG\OG-Technical Services Team
                    ITCG\AC-RD Access - Dev
                    ITCG\LG-HQ Users
                    ITCG\AC-VPN Users
                    ITCG\AC-RD Access - Farm 2
                    ITCG\AC-RD Access - Farm 1
                    Authentication authority asserted identity
                    ITCG\RC-Accounting
                    ITCG\PR-NP03
                    ITCG\RC-ITCG Internal
                    ITCG\RC-ITCG
                    ITCG\RC-Multimedia-II
                    ITCG\RC-Development
                    ITCG\RC-Photos
                    ITCG\RC-MSDN
                    ITCG\PR-NP01
                    ITCG\RC-eBooks
                    ITCG\RC-Development-Kix
                    ITCG\RC-Development-Internal
                    ITCG\RC-Multimedia
                    ITCG\RC-SWDist
                    ITCG\RC-WebShare
                    Mandatory Label\Medium Mandatory Level
                    RC-Backup
                    Users


 - ENVIRONMENT START -
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\glenn\AppData\Roaming
CommonProgramFiles=C:\Program Files (x86)\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=IHWD001
ComSpec=C:\Windows\system32\cmd.exe
FP_NO_HOST_CHECK=NO
HOMEDRIVE=H:
HOMEPATH=\
HOMESHARE=\\ifpsp01\Users\glenn
KixLibPath=K:\KixLib
LOCALAPPDATA=C:\Users\glenn\AppData\Local
LOGONSERVER=\\IHWIADCP01
NUMBER_OF_PROCESSORS=8
OS=Windows_NT
Path=C:\PROGRA~2\ITCG\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C
:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\Windows System
Resource Manager\bin;;C:\Program Files (x86)\jZip;C:\Program Files
(x86)\Microchip\MPLAB C32 Suite\bin;C:\Program Files (x86)\Microchip\MPLAB
IDE\VDI;C:\Program Files (x86)\HI-TECH Software\PICC\PRO\9.65\bin;C:\Program Files
(x86)\Windows Live\Shared
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.KIX;.KX;.KXW;.KW
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_ARCHITEW6432=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 30 Stepping 5, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=1e05
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files (x86)
ProgramFiles(x86)=C:\Program Files (x86)
```

```
ProgramW6432=C:\Program Files
PROMPT=$CIHWD011$F - $P$G
PSModulePath=C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
PUBLIC=C:\Users\Public
SESSIONNAME=RDP-Tcp#0
SUID=ITCG\executive
SystemDrive=C:
SystemRoot=C:\Windows
S_BIN=C:\PROGRA~2\ITCG\bin
S_CONFIG=C:\PROGRA~2\ITCG
S_LIB=C:\PROGRA~2\ITCG\Lib
S_LOGS=C:\PROGRA~2\ITCG\Logs
TEMP=C:\Users\GLENN\AppData\Local\Temp
TMP=C:\Users\GLENN\AppData\Local\Temp
USERDNSDOMAIN=ITCG.PVT
USERDOMAIN=ITCG
USERDOMAIN_ROAMINGPROFILE=ITCG
USERNAME=gbarnas
USERPROFILE=C:\Users\glenn
windir=C:\Windows
 - ENVIRONMENT END -





Processing DATA_Fam_F (DISK)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group RC-Family - continuing.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ifpsp01\family
 DESC Translated: Family Data


Processing DATA_CORP (DISK)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group RC-ITCG - continuing.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ifpsp01\itcg
 DESC Translated: Shared Data


Processing ACCOUNTING (DISK)
 Eval: 13, Man: 8, IM: 4, IA: 0, Chk: 1
  Member of mandatory excluded group RC-ITCG - exiting.


Processing HOME (DISK)
    OK to process this resource!
 PATH Translated: HOME
 DESC Translated: Personal Folder


Processing MSDN (DISK)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group RC-MSDN - continuing.
 Eval: 0, Man: 0, IM: 0, IA: 0, Chk: 0
  No match in allowed group RC-MSDN_RO - ignored.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ifpsp01\MSDN
 DESC Translated: MSDN Library


Processing eBOOKS (DISK)
    OK to process this resource!
 Creating UNC shortcut resource at TID 9.
 PATH Translated: \\ifpsp01\eBooks
 DESC Translated: eBook Library
 Shortcut creation Enabled.


Processing DEV (DISK)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group RC-Development - continuing.
 Eval: 0, Man: 0, IM: 0, IA: 0, Chk: 0
  No match in allowed group Domain Admins - ignored.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ifpsp01\Dev
 DESC Translated: Software Dev


Processing AUDIO (DISK)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group RC-Multimedia - continuing.
 Eval: 0, Man: 0, IM: 0, IA: 0, Chk: 0
  No match in allowed group RC-Multimedia_RO - ignored.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ismsp01\Audio
 DESC Translated: Audio Library


Processing MOVIES (DISK)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group RC-Multimedia - continuing.
 Eval: 0, Man: 0, IM: 0, IA: 0, Chk: 0
  No match in allowed group RC-Multimedia_RO - ignored.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ismsp01\Movies
 DESC Translated: Movie Library


Processing MATURE MOVIES (DISK)
 Eval: 0, Man: 0, IM: 0, IA: 0, Chk: 0
  No match in allowed group RC-Multimedia-Mature - ignored.


Processing PHOTOS (DISK)
    OK to process this resource!
 Creating UNC shortcut resource at TID 26.
 PATH Translated: \\ismsp01\photos
 DESC Translated: Photo Library
 Shortcut creation Enabled.


Processing MEDIA ROOT (DISK)
  Member of allowed User List - continuing.
    OK to process this resource!
 Creating UNC shortcut resource at TID 27.
 PATH Translated: \\ismsp01\MediaRoot
 DESC Translated: Media Library
 Shortcut creation Enabled.


Processing SWDIST (DISK)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group RC-SWDist - continuing.
 Eval: 0, Man: 0, IM: 0, IA: 0, Chk: 0
  No match in allowed group AC-Workstation Admins - ignored.
 Eval: 0, Man: 0, IM: 0, IA: 0, Chk: 0
  No match in allowed group AC-Server Admins - ignored.
 Eval: 0, Man: 0, IM: 0, IA: 0, Chk: 0
  No match in allowed group Domain Admins - ignored.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ifpsp01\SWDIST
 DESC Translated: SWDIST Install Point


Processing MOTD_General (MESSAGE)
    OK to process this resource!
 PATH Translated: \\ifpsp01\Users\MOTD.txt
 DESC Translated: Message of the day


Processing MOTD_Admins (MESSAGE)
  requires ADMIN privilege - skipping.


Processing HP4350PCL (PRINT)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group PR-NP01 - continuing.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ifpsp01\ihphpm01
    New - Slot 0: HP4350PCL


Processing BR6710 (PRINT)
 Eval: 1, Man: 0, IM: 0, IA: 0, Chk: 1
  Member of allowed group PR-NP03 - continuing.
 Logic: AND, Value: 3, Required: 3
    OK to process this resource!
 PATH Translated: \\ifpsp01\ihpbrc01
    New - Slot 1: BR6710

Processing printer default settings
 Printer \\ifpsp01\ihphpm01: default is soft-set
 Printer \\ifpsp01\ihphpm01: has Default Eligible status

Processing active resources:
   Disk: F   \\ifpsp01\family
   Disk: G   \\ifpsp01\itcg
   Disk: H   HOME
   Disk: I   \\ifpsp01\MSDN
   Disk: J   \\ifpsp01\eBooks
   Disk: K   \\ifpsp01\Dev
   Disk: M   \\ismsp01\Audio
   Disk: N   \\ismsp01\Movies
   Disk: Y   \\ifpsp01\SWDIST
Processing Shortcut Only records
   Disk: UNC \\ismsp01\photos
   Disk: UNC \\icwismsp01\MediaRoot
```

```
Printer:      \\ifpsp01\ihphpm01  S
              SoftMap - Exists
              Pre-existing default
Printer:      \\ifpsp01\ihpbrc01  N
              SoftMap - Exists
Message:      \\ifpsp01\Users\MOTD.txt  [0.1]
```

The header shows the date & time that the entry occurred as well as some information about the computer, group membership, and the user environment. Following that are the resource records that were processed and evaluated. The data on the MVLookup lines represent what was read from the config file, while the Translated lines represent the result of any Value Rewrite process. Both the PATH and the DESC lines are processed by MVLookup.

The Test8 resource record has some more complex parameters for Group or OU processing. The result of the individual evaluations is written to the log in an Eval line.

`Processing TEST8` <– the name of the resource record is identified

```
 Eval: 12, Man: 8, IM: 4, IA: 0, Chk: 0
```

These are the flags defined by resource processing.

EVAL is the sum of the other 4 flags.

Man is the value of the Mandatory Membership flag, either 0 or 8. When set to 8, it indicates that the user must be a member of all named groups.

IM is the value of the Inverse Membership flag, either 0 or 4. When set to 4, it indicates that the user must NOT be a member of the named group.

IA is the value of the Inverse Action flag, either 0 or 2. When set to 2, it inverts the resource action. That is, if the user is a member of the named group, the action that would normally occur will not. This option is not regularly used, but is available for special situations.

Chk is the value of the comparison flag, and it indicates that the user has been authorized to use the resource at a basic level.

Below each Eval: line will be the result status, which describes the conclusion of the logic process, and the action that will be taken. The action is one of:

Ignored – An access restriction is defined but the current evaluation returned neither an Allow nor Deny result, so that restriction will be ignored. Further processing will occur to determine if access to the resource will be permitted.

Continuing – The current evaluation returned an Allow result. The resource will be permitted as long as any further evaluations do not return a Deny result.

Exiting – The current evaluation returned a Deny result. Access to the resource will be denied and no further processing will occur

If no restrictions were specified, or an Allow condition was met when restrictions were specified without a subsequent Deny condition, the message “OK to map this resource” will appear.

The last section of the log lists the resource records that are being processed. It shows the resource type, target (where appropriate), and path.

Diagnosing Problems

To diagnose resource connection issues, make sure that a LoginDebug.log file is generated, either by creating the empty file or running the login script with the `--d` parameter.

Look at the end of the lot to review the mappings that were performed. This lists the resource type, target ID (drive letter), and path. The two most common issues are incorrect processing or no processing.

Incorrect Processing – a resource is mapped, but not the desired resource. This often occurs when multiple resource records share a common target.

- You will need a copy of the config file. Start by locating the resource that was incorrectly processed and identify the Target value.
- Locate the desired resource record and identify the Target value. It should match the Target in the incorrect resource record.
- Review the LoginDebug.log file – verify that the desired resources has an "OK to process this resource" message.
    - If there is no "OK to process" message, troubleshoot following the No Processing method that follows.
    - If the "OK to process" message is present, the most likely issue is that the desired resource is defined before the resource that is processed. This can be resolved by changing the sequence of resource records so they appear in the least to most desirable order, or by using the PRIORITY parameter. Add a PRIORITY=5 to the resource that is preferred, or use Priority values on all resources that share a common target.

No Processing – the desired resource is not processed. This is generally the result of specifying a process control that is not met, such as a group or OU membership. For Message class resources, the defined message file may not be present.

- Using a copy of the config and LoginDebug.log files, review the controls associated with the resource.
    - If OUS have been defined:
        - Locate the "User OU" value from the LoginDebug.log file. Make sure it exists in the OUS defined in the config file. Verify that the spelling is correct!
    - If GROUPS have been defined in the resource:
        - Compare the groups in the resource record with those listed in the Group Membership section of the LoginDebug.log file.
        - If no match is found, make sure that the user is a member of the required groups. The user may need to be a member of multiple groups if Mandatory membership is enabled. Also, make sure that the group name and Windows 2000 Compatibility names are the same! A common problem occurs when a group is created with an incorrect

name. The group is then renamed, but the compatibility name retains the original, non-matching value.

- Review the LoginDebug.log file and examine the results of the resource in question. If an “exiting” message is displayed, review the associated group setting.
- If the user was recently added to the AD Group, the old group data may still be cached by Kixtart. You can manually flush the group cache via two simple methods:
    - Run `Kix32.exe /f` to flush the cache.
    - Delete HKCU\Software\KiXtart\TokenCache from the registry. The *Universal Login Script* will automatically delete the token cache every 7 days. The cycle can be adjusted by editing the FlushTokenCache parameter in the COMMON section of the configuration file.

Level 2 Debugging

Level 2 debugging is enabled by creating a file called *debug.txt* in the same folder where the login script was executed from. During level-2 debugging, no resources are actually mapped, and no commands are executed. The commands used to perform the resource actions are displayed/logged so they can be verified prior to use in a production deployment. A *LoginDebug.log* file is created automatically when Level 2 debugging is active.

Error Logging

If any errors occur during login processing, error messages are written to the %UserProfile%\loginerrs.log file. Resources that fail to map are displayed in red. If message output is suppressed and errors occur, a window will open with the error messages displayed in red.

Any time an error message is displayed during login, the window will remain open for 15 minutes or until a key is pressed. The Break setting is also enabled to allow the user to close the login script window without causing a logoff. This provides enough time for a user to see that an error occurred and contact the help desk with an appropriate message. If the user closes the error screen, the help desk technician can review the *LoginErrs.log* file and the *LoginDebug.log* file (if present) for details of the error.

Performance Analysis

The login script can write timestamps to a performance log to help identify performance related issues. Process data is written to %UserProfile%\LoginPerf.log with date and time stamps accurate to 1ms. A timestamp is written when the script starts, at the start of each significant block of code, at the start and end of each resource decision, and when the script completes. Performance analysis is only available when the login script is executed from the NetLogon share.

Timestamps are enabled by creating a file called "LSPERF.TXT" in the root of the NetLogon share. When timestamp logging is enabled, the version number displayed during logon will display a "_ts" suffix, similar to "2.8.0b_ts".

A typical configuration that evaluates 10-15 drive resources, 6-8 printer resources, and a small number of message and command resources will usually complete in less than 4 seconds. Larger configurations may take 3-9 seconds to complete. If your login configuration is taking more than 10 seconds to complete (exclusive of external script processing), performance logging should be enabled to determine where the delay is occurring. This is quite often COMMAND resources loading and executing. The log will report the following for a COMMAND process:

```
2012/05/02 20:31:00.558 - COMMAND Start: \\DOM\NetLogon\Kix32.exe \\DOM\NetLogon\NoHidden.kix
2012/05/02 20:31:00.605 -  - Complete
```

This shows that the command took 47ms to complete.

Note that performance logging will add a minimal amount of processing and disk I/O to the login process – usually less than 50ms. Timestamps have an accuracy of 16ms (+/- 8ms).

Performance Logging Example

Since this example login process enabled performance logging, let’s look at the resulting file. The log contents that follow illustrate how the script runs through the authorization and mapping process, checking a total of 22 resource definitions in just 2.533 seconds.

```
2013/03/12 15:51:34.071 - Start Login Script
2013/03/12 15:51:34.087 - INI Defined
2013/03/12 15:51:34.144 - License check complete
2013/03/12 15:51:34.228 - Display Start
2013/03/12 15:51:34.328 - Init Complete
2013/03/12 15:51:34.328 - Clear Drive Mappings
2013/03/12 15:51:34.409 - Removing drive F:
2013/03/12 15:51:34.454 -  - Complete
2013/03/12 15:51:34.454 - Removing drive G:
2013/03/12 15:51:34.499 -  - Complete
2013/03/12 15:51:34.500 - Removing drive I:
2013/03/12 15:51:34.542 -  - Complete
2013/03/12 15:51:34.542 - Removing drive J:
2013/03/12 15:51:34.584 -  - Complete
2013/03/12 15:51:34.584 - Removing drive K:
2013/03/12 15:51:34.627 -  - Complete
2013/03/12 15:51:34.628 - Removing drive N:
2013/03/12 15:51:34.667 -  - Complete
2013/03/12 15:51:34.667 - Removing drive P:
2013/03/12 15:51:34.709 -  - Complete
2013/03/12 15:51:34.710 - Removing drive Y:
2013/03/12 15:51:34.753 -  - Complete
2013/03/12 15:51:34.754 - Removing drive
2013/03/12 15:51:34.804 -  - Complete
2013/03/12 15:51:34.804 - Removing drive
2013/03/12 15:51:34.857 -  - Complete
2013/03/12 15:51:34.858 - Removing drive
2013/03/12 15:51:34.925 -  - Complete
2013/03/12 15:51:34.925 - Removing drive
2013/03/12 15:51:34.982 -  - Complete
2013/03/12 15:51:34.983 - Removing drive
2013/03/12 15:51:35.035 -  - Complete
2013/03/12 15:51:35.042 - Checking COMMON
2013/03/12 15:51:35.045 -  - Complete
2013/03/12 15:51:35.109 - Checking GROUP_BASED_CONFIG
2013/03/12 15:51:35.112 -  - Complete
2013/03/12 15:51:35.117 - Checking DATA_F
2013/03/12 15:51:35.149 -  DISK Resource
2013/03/12 15:51:35.159 -  - Complete
2013/03/12 15:51:35.163 - Checking DATA_CORP
2013/03/12 15:51:35.192 -  DISK Resource
2013/03/12 15:51:35.201 -  - Complete
2013/03/12 15:51:35.205 - Checking DATA_CORP_Test
2013/03/12 15:51:35.208 -  - Complete
2013/03/12 15:51:35.212 - Checking ACCOUNTING
2013/03/12 15:51:35.251 -  - Complete
2013/03/12 15:51:35.256 - Checking HOME
2013/03/12 15:51:35.268 -  DISK Resource
2013/03/12 15:51:35.279 -  - Complete
2013/03/12 15:51:35.284 - Checking MSDN
2013/03/12 15:51:35.336 -  DISK Resource
2013/03/12 15:51:35.345 -  - Complete
2013/03/12 15:51:35.351 - Checking KixDev
2013/03/12 15:51:35.382 -  DISK Resource
2013/03/12 15:51:35.391 -  - Complete
2013/03/12 15:51:35.395 - Checking Dev
2013/03/12 15:51:35.430 -  DISK Resource
2013/03/12 15:51:35.440 -  - Complete
2013/03/12 15:51:35.445 - Checking MUSIC0
2013/03/12 15:51:35.462 -  - Complete
2013/03/12 15:51:35.466 - Checking PHOTOS
2013/03/12 15:51:35.499 -  DISK Resource
2013/03/12 15:51:35.509 -  - Complete
2013/03/12 15:51:35.516 - Checking SWDIST
2013/03/12 15:51:35.529 -  DISK Resource
2013/03/12 15:51:35.539 -  - Complete
2013/03/12 15:51:35.543 - Checking MOTD_General
2013/03/12 15:51:35.558 -  MESSAGE Resource
2013/03/12 15:51:35.568 -  - Complete
2013/03/12 15:51:35.573 - Checking MOTD_Admins
2013/03/12 15:51:35.585 -  MESSAGE Resource
2013/03/12 15:51:35.597 -  - Complete
2013/03/12 15:51:35.602 - Checking MOTD_All
2013/03/12 15:51:35.620 -  - Complete
2013/03/12 15:51:35.625 - Checking MOTD_Work
2013/03/12 15:51:35.647 -  - Complete
2013/03/12 15:51:35.652 - Checking HP4350PCL
2013/03/12 15:51:35.682 -  PRINT Resource
```

```
2013/03/12 15:51:35.693 -  - Complete
2013/03/12 15:51:35.698 - Checking BR6710
2013/03/12 15:51:35.717 -  - Complete
2013/03/12 15:51:35.722 - Checking NoHiddenLogon
2013/03/12 15:51:35.749 -  COMMAND Resource
2013/03/12 15:51:35.754 -  COMMAND: Path Translate
2013/03/12 15:51:35.755 - COMMAND: Args
2013/03/12 15:51:35.764 -  COMMAND: Complete!
2013/03/12 15:51:35.764 -  - Complete
2013/03/12 15:51:35.764 - Printer Prep Start
2013/03/12 15:51:35.780 -  - Complete
2013/03/12 15:51:35.785 - Processing Drive A:
2013/03/12 15:51:35.786 -  - Complete
2013/03/12 15:51:35.786 - Processing Drive B:
2013/03/12 15:51:35.787 -  - Complete
2013/03/12 15:51:35.787 - Processing Drive C:
2013/03/12 15:51:35.787 -  - Complete
2013/03/12 15:51:35.788 - Processing Drive D:
2013/03/12 15:51:35.788 -  - Complete
2013/03/12 15:51:35.788 - Processing Drive E:
2013/03/12 15:51:35.789 -  - Complete
2013/03/12 15:51:35.789 - Processing Drive F:
2013/03/12 15:51:35.897 -  - Complete
2013/03/12 15:51:35.898 - Processing Drive G:
2013/03/12 15:51:36.021 -  - Complete
2013/03/12 15:51:36.022 - Processing Drive H:
2013/03/12 15:51:36.029 -  - Complete
2013/03/12 15:51:36.029 - Processing Drive I:
2013/03/12 15:51:36.129 -  - Complete
2013/03/12 15:51:36.129 - Processing Drive J:
2013/03/12 15:51:36.228 -  - Complete
2013/03/12 15:51:36.228 - Processing Drive K:
2013/03/12 15:51:36.285 -  - Complete
2013/03/12 15:51:36.285 - Processing Drive L:
2013/03/12 15:51:36.286 -  - Complete
2013/03/12 15:51:36.286 - Processing Drive M:
2013/03/12 15:51:36.286 -  - Complete
2013/03/12 15:51:36.287 - Processing Drive N:
2013/03/12 15:51:36.287 -  - Complete
2013/03/12 15:51:36.288 - Processing Drive O:
2013/03/12 15:51:36.288 -  - Complete
2013/03/12 15:51:36.288 - Processing Drive P:
2013/03/12 15:51:36.391 -  - Complete
2013/03/12 15:51:36.391 - Processing Drive Q:
2013/03/12 15:51:36.392 -  - Complete
2013/03/12 15:51:36.392 - Processing Drive R:
2013/03/12 15:51:36.393 -  - Complete
2013/03/12 15:51:36.393 - Processing Drive S:
2013/03/12 15:51:36.394 -  - Complete
2013/03/12 15:51:36.394 - Processing Drive T:
2013/03/12 15:51:36.395 -  - Complete
2013/03/12 15:51:36.395 - Processing Drive U:
2013/03/12 15:51:36.396 -  - Complete
2013/03/12 15:51:36.396 - Processing Drive V:
2013/03/12 15:51:36.397 -  - Complete
2013/03/12 15:51:36.397 - Processing Drive W:
2013/03/12 15:51:36.397 -  - Complete
2013/03/12 15:51:36.398 - Processing Drive X:
2013/03/12 15:51:36.398 -  - Complete
2013/03/12 15:51:36.399 - Processing Drive Y:
2013/03/12 15:51:36.504 -  - Complete
2013/03/12 15:51:36.504 - Processing Drive Z:
2013/03/12 15:51:36.505 -  - Complete
2013/03/12 15:51:36.506 - Processing Printer 0
2013/03/12 15:51:36.523 -  - Complete
2013/03/12 15:51:36.643 - COMMAND Start: Kix32.exe \\CORP\NetLogon\NoHiddenLogon.kix
2013/03/12 15:51:36.696 -  - Complete
2013/03/12 15:51:36.697 - Login Script Complete
```

Configuration Examples

The supplied *login.ini* file is rich with examples, from the basic to the very complex. It contains comments to help you understand the parameters that were selected. The examples here will walk you through some configuration scenarios with a “problem solving” approach.

Basic Configuration Scenarios

Process a Resource for All Users

The most basic mapping – all users that run the login script will map this resource.

```
[AllUsers]
CLASS=Disk
PATH=\\server\share
TARGET=G
```

With no authorization parameters present, every user will map this resource. The only thing you might add to this is a DESC parameter to provide a description!

Display Alert for All Admins

A message file will be displayed for all admin-level logins to remind the user to proceed with caution. Red text is used for emphasis, and a 5-second delay is defined.

```
[AllAdminMessage]
CLASS=Message
PATH=\\server\share\Message
DESC=Admin Alert!
COLOR=Red
DELAY=5
PRIV=Admin
```

Process a Resource for Members of One or More Groups

A common task – map a resource based on group membership. In this case, all members of either the Finance or Accounting groups will map the drive.

```
[FinanceShare]
CLASS=Disk
PATH=\\server\share
TARGET=G
GROUPS=Finance,Accounting
```

Process a Resource for Multiple Group Membership

This will process a resource when a user is a member of two or more groups, allowing complex authorization methods. Very similar to the previous example, the addition of a leading “+” to the GROUPS list will require membership in *all* of the listed groups instead of *any* of the listed groups.

```
[FinanceShare]
CLASS=Disk
PATH=\\server\share
TARGET=G
GROUPS=+,Finance,Accounting
```

Process a Resource via AD Attribute

A common need but often difficult – map a resource based on a user being in the Accounting Department according to their AD Attribute, not Group or OU membership.

```
[FinanceShare]
CLASS=Disk
PATH=\\server\share
TARGET=G
ADATTR=Department:Accounting
```

Process a Resource Based on OU

OU-based mapping examines the entire DN string for the current user. A DN string looks like this: `CN=smith\, John,OU=department 3,OU=Users,DC=Fabricam,DC=com`. The OU matching logic ignores the CN= and DC= components and examines just the OU= components. If any of the OUs listed are found in an OU= component of the DN string, it is considered a match. The OU must match the exact “OU=name,” part of the string to be valid, including the leading “OU=” and the trailing comma. This example maps the resource for objects contained in either of the HelpDesk or IT OUs.

```
[TechServices]
CLASS=Disk
PATH=\\server\share
TARGET=T
OUS=HelpDesk,IT
```

Process a Resource Based on Group and OU

In this example, a user must be a member of the Accounting group, and should be in the Central Division OU. The Accounting group contains members from East, Central, and West Division OUs. By specifying both Group and OU parameters, we can restrict the resource to just members of the Accounting group located in the Central Division.

```
[Central Accounting]
CLASS=Disk
PATH=\\server\share
TARGET=F
OUS=Central Division
GROUPS=Accounting
```

If we wanted to map the resource if a user was a member of the Accounting group *or* the Central Division OU, we could add the following line to this resource record:

```
LOGIC=OR
```

This changes the comparison of Group, OU, and AD Attribute parameters from an AND condition (all requirements must be met) to an OR condition (either requirement is met).

Displaying a Message During Logon

A message contained in a text file can be displayed during logon. The file can be on a mapped drive or specified as a UNC path. The file *format* must be plain text. Each message is terminated with a line of dashes, and a default delay of 2 seconds is performed unless a specific delay is defined. The color of the text for the message body can be specified, and a short header can be defined using the DESC parameter.

```
;Message for All Users
[MessageOfTheDay]
CLASS=MESSAGE
PATH=\\server\share\MOTD.txt
DESC=Message of the Day
COLOR=Yellow

;Message for Admin Users
[AdminWarning]
CLASS=MESSAGE
PATH=AdminWarning.txt
PRIV=admin
DESC=Administrator Access
COLOR=Red
```

If the PATH value is not rooted to a specific location, the script will check the directory where the login script started and then the root of the NetLogon share (if different). This allows message files to be placed on the NetLogon share rather than a specific server/share.

Message processing occurs after all drives are mapped so that mapped resources can be used to reference the message text files.

Note that the second example uses a non-rooted file in the PATH. This will be found on the NetLogon or Startup folder. The PRIV parameter restricts this message to users with local administrative access, and displays in red to get their attention.

Running a Command

Commands are often run during logon to customize the user environment. Depending on the level of local access, many different types of tasks can be performed. The example below runs a Kixtart script that prompts the user to choose one of 3 available and authorized screen saver formats. The Kix32.exe in the PATH parameter is non-rooted and will be run by default from the NetLogon share. The argument to Kix32.exe is the fully-qualified path to the script, including any script arguments. This uses the Shell method to invoke a new Kix32 instance to insure isolation from the login script environment.

```
[SetScreenSaver]
CLASS=COMMAND
PATH=Kix32.exe
ARGS=\\%USERDOMAIN%\NetLogon\UserScreenSaver.kix
```

A similar form is shown below. Using the Call method, it uses the login script instance of Kix32 for faster loading and execution. The Kixtart script was validated to not interfere with any of the global variables or UDFs in the login script.

```
[SetScreenSaver]
CLASS=COMMAND
METHOD=Call
PATH=\\%USERDOMAIN%\NetLogon\UserScreenSaver.kix
```

Acceptable Use Policy Message

Companies often wish to display a message defining the acceptable use of the computer and network resources during logon, and force a logoff if the user does not accept the policy. This can be accomplished using a MESSAGE resource and defining a PROMPT parameter. While the message file *must* exist, it can be empty or simply a “please review & accept the acceptable use policy to continue your access.”

The following configuration displays a popup message that will log the user off if they do not accept the policy. The UsePolicy.txt is an empty file in the root of the NetLogon share.

```
[Use Policy]
CLASS=MESSAGE
PATH=UsePolicy.txt
PROMPT=1;Acceptable Use Policy; legal notice here…\n\nDo you accept these
terms?;;LOGOFF
```

Note that the PROMPT text must all be on one line – it is wrapped here to fit the page. The “legal notice here…” would be replaced with up to 1024 characters containing your acceptable use policy. No action is needed for a Yes response, so the field between the message and NO_Action is empty. The NO_Action is “Logoff”, which forces an immediate logoff of the user. At this time, the only available action is “Logoff”.

You can use the “\n” in your message text to force a line break at that point. This is illustrated in the example above.

Advanced Configuration Scenarios

Configuring Branch Office or Department Shares with Value Rewrite

Assumptions:

- Each branch office or department is defined by a unique OU
- The OUOffset is properly defined based on the configuration of your DN string.
- Branch offices have specific share folders (\\server\shares\Dept_12)
- Branch office shares all map their primary share to the same drive letter (S:)

The first step is to define the share entry in the configuration file that uses Path Rewriting:

```
[DeptShare]
CLASS=DISK
TARGET=S
PATH=&OU:DeptMap&
```

<- this is the path rewrite entry, defining the DeptMap record.

A set of mapping records is also required. If you have already configured a department share and mapping records, and are simply adding mapping for a new department, the resource record and mapping record will already exist. Identify the Path Rewrite section and just add a record for the new department to it.

```
[DeptMap]
Dept 11=\\server\shares\Dept_11
Dept 12=\\server\shares\Dept_12
```

This configuration will identify the OU that a user is a member of, search the mapping record table for a match, and connect the share that was found (if any) to the target drive letter.

The first time you create a rewrite map table, you need to enter all of the department OU names. Later, as new departments or branch offices are added, you simply add a new entry to the rewrite map table, linking the share resource to the department's OU.

Note: The rewrite table does not need to be an entire UNC path. If you have multiple file servers with identical shares and resources, the lookup can be simply the server name, which would allow the same lookup table to be used for several paths being rewritten. Similarly, you might have one file server with several department-specific shares. This is especially useful with Site or Subnet rewrites. For example:

```
[DeptShare]                     [DeptShare]
CLASS=DISK                      CLASS=DISK
TARGET=S                        TARGET=S
PATH=&OU:DeptMap&\share         PATH=\\server\&OU:DeptMap&

[DeptMap]                       [DeptMap]
Dept 11=\\server11              Dept 11=Dept_11
Dept 12=\\server12              Dept 12=Dept_12
```

The left column illustrates basic server name substitution while the right column illustrates share name substitution.

Adding a Second Department to an Existing Share

Another common scenario is when a “parent” department needs access to “child” departments. One example might be that the “Accounting” OU members need access to their “Accounting Dept” share, but also the “Receivables” share that the AR department uses.

Locate the resource record for the Receivables share:

```
[ARDeptShare]                          [ARDeptShare]
CLASS=DISK                             CLASS=DISK
TARGET=S                               TARGET=S
PATH=\\server\receivables              PATH=\\server\receivables
OU=AR Dept                             OU=AR Dept,Accounting Dept
```

Add the “,Accounting Dept” OU name to the list of OUs that can access this share. Once this is done, users of both departments will be able to utilize this drive mapping after the next logon.

Using Configuration Sets

Configuration sets allow a single configuration file to support the needs of multiple regions or departments without depending on OU structure or group membership. The name of the configuration set is specified on the command line when the login script is executed. Only resource records with matching CFGSET parameters are executed, along with resource records that do not have a CFGSET parameter. These are considered global or common resources and will map for all users.

Start by creating the resource records used by all users. Recognize that some of these may be overridden by CFGSET-specific records.

Add the CFGSET-specific records just like any other, but include the CFGSET value.

```
[DeptShare]
CLASS=DISK
TARGET=S
PATH=\\server\share
CFGSET=MARKETING
PRIORITY=5
```

The priority insures that this record will take precedence over any common records, regardless of the sequence of the records in the config file. When the login script is invoked with

```
Kix32.exe Kixtart.kix --c MARKETING
```

all of the common records and records with CFGSET=MARKETING will be processed, ignoring all other records that have different CFGSET values. All Value Rewrite and other features are supported as expected.

Alternate or Sub-Folder Mapping

Consider the following situation – the finance team has access to a share that contains the accounting data folder as well as other subfolders for documents & correspondence, spreadsheets, and other corporate data. The external accounting firm should be able to access the accounting data but not the other data files. Both regular users and the external accountants will map the data to the G: drive. In this case we will use two Resource Records and employ Mandatory group membership.

```
; Map finance data share to G: if a member of the "finance" group
[FIN_DATA_CORP]
CLASS=DISK
TARGET=G:
PATH=\\fpsp01\FinanceData
DESC=Shared Finance Data
GROUPS=AC-Finance

; Map the accounting folder to the G: drive for accountant access
[FIN_DATA_ACCOUNTANT]
CLASS=DISK
TARGET=G:
PATH=\\fpsp01\FinanceData\accounting
DESC=Accounting Data
GROUPS=+,!AC-Finance,AC-Accountants
```

The first record maps the resource root to the G: drive for members of the AC-Finance group. When a member of AC-Finance logs in, the first record is permitted and added to the process list. When the second record is evaluated it is determined that mandatory group membership is enabled (leading “+”). Negation is applied to the AC-Finance group, so members of the AC-Accountants group who are NOT members of the AC-Finance group will be permitted. The use of negation insures that internal finance team members process the first resource and ignore the second. External accountants, being members of AC-Accountants and not AC-Finance, process the second record.

Gross Access Filtering – Admin, Guest, or NoGuest

This example covers a less-common situation. The art department has full-time and freelance graphic artists. All are members of the AC-Media group with access to other media shares. One of the shares contains the photos of employees used for security badges and corporate events. These photos should be restricted only to authorized employees. The freelance artists are members of Domain Guests instead of Domain Users, restricting their rights outside of the Media shares. We can take advantage of this by allowing access to the employee photo share to members of the AC-Media group, but restrict access from anyone with Guest level access. The configuration below permits only the full-time employees of the AC-Media group to access the share.

```
[EMPLOYEE_PHOTOS]
CLASS=DISK
TARGET=P:
PATH=\\fpsp01\ephotos
DESC=Employee Photo Library
PRIV=NoGuest
GROUPS=AC-Media,domain admins
PRIORITY=1
```

Compound Value Rewrites

Compound rewriting is a powerful concept that significantly extends the resource mapping process. For example, you might have several departments that need access to departmental shares. If each department is represented by an OU, this becomes a fairly simple OU:table type lookup. If, however, you need to change the server that you connect to based on the network subnet that the user is in, a compound, or recursive lookup can be employed.

The simple vs. compound / recursive lookups would be:

```
[DeptShare]                      [DeptShare]
CLASS=DISK                       CLASS=DISK
TARGET=P                         TARGET=P
PATH=&OU:DeptByOU&               PATH=&OU:DeptByOU&

[DeptByOU]                       [DeptByOU]
OU1=\\server\share1              OU1=&SUBNET:Dept1BySubnet&\share1
OU2=\\server\share2              OU2=&SUBNET:Dept2BySubnet&\share2

                                 [Dept1BySubnet]
                                 192.168.32.0=\\server1
                                 192.168.34.0=\\server2

                                 [Dept2BySubnet]
                                 192.168.32.0=\\server1
                                 192.168.34.0=\\server2
```

In the first example, members of OU1 are simply mapped to \\server\\share1. In the second example, members of OU1 perform a lookup that points to a department specific subnet lookup. The ByDeptOU defines the share name but requires a recursive lookup to define the server. The second lookup sets the server name. Thus, if a user is in OU1 and in the 192.168.34.0 network, their resource would be mapped to \\server2\share1.

Group Based Config-File Selection

This offers a unique ability to automatically select a configuration file based on Active Directory group membership. This capability can impact performance of the script if not properly configured and thus deserves some special attention. Due to the additional overhead of reading multiple configuration files, it is recommended that this feature be used to facilitate migration and consolidation efforts rather than permanent production use.

In most cases, lookup tables and Value Rewrite will provide the flexibility required for most complex configurations. Beyond this, placing the script and config file into specific folders and defining the login script as "kix32.exe Folder\kixtart.kix" can easily accommodate regional script requirements with unique configuration files. Similarly, user profiles can define specific configuration files with the "`--i filename`" or the "`--c CfgSetID`" parameters.

When the other options don't offer the flexibility, the automatic configuration file selection method allows the script to dynamically switch the configuration file when the user is a member of a specific group. This can simplify the login script configuration settings in the user profile, eliminating any arguments and maintaining a flat NetLogon folder structure. To effectively implement this, some important guidelines should be followed.

- **AD Groups** – the groups should be used specifically to define which login script configuration to use. These groups should not contain other nested groups, and users should only be a member of a single group that defines the login configuration. Should a user be a member of multiple groups, only the first one matched will be used, and all others ignored. This may be in conflict with the recommendation below. Using generic groups or allowing multiple group membership will lead to confusion and potentially incorrect resource processing.
- **Performance on Slow Links** – The login script is designed to run efficiently even on slow (64Kbps or dialup) links through configuration file caching. Since this logic controls which configuration file will be used, the queries needed to make this decision will be made to the network (netlogon) copy of the configuration file and not the cached copy. If a large number of queries must be made, the login performance will be affected. This can be minimized by placing the definitions with slower connection speeds first in the lookup list, minimizing the number of queries to make over the slow links. This will not have significant impact on faster WAN links or LAN connections.
- **Sequence of Configuration File Selection** – the script starts by making "login.ini" the active config file. It then checks for a configuration file specified on the command line, a config file matched by group membership, a computer-specific file, and finally a user-specific file.

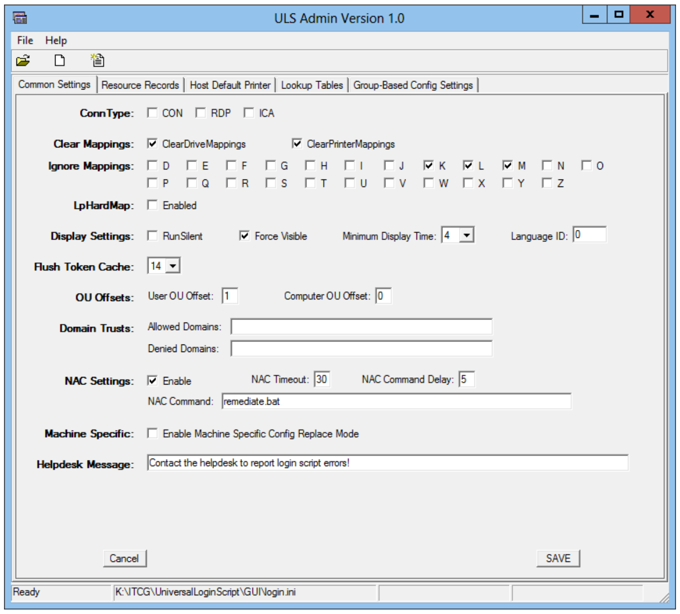
  The process insures that a configuration file is always available for use, starting with the most generic to the most specific. If the "login.ini" file is not present, the script will exit gracefully with an error in the log.

GUI Administration Console

The GUI administration console provides a simplified interface for editing the ULS configuration files. The console allows any authorized user to open a configuration file and dynamically change the configuration settings without any programming or even INI file editing experience.

The current revision of the admin console allows control of all COMMON settings, all standard Resource settings, and basic editing of all lookup tables (standard and user-defined). The user-defined (USER_*name*) Resource Record parameters are free-form and thus require the user to edit the configuration file directly. The ability to query AD for lists of users, groups, OUs, and Sites (used on the Resources tab) is planned for a future release. These objects must currently be entered manually, as they would be if the configuration file was edited directly.

The ULS Admin Console main screen is shown here. All input is locked until a config file is loaded or created. Once the admin console is associated with a configuration file, any changes made on the screen will update the file immediately upon clicking the Save button. Any change will set a MODIFIED flag, displayed in the bottom-right of the status bar, and will prevent the user from switching tabs. A warning will be displayed if the user tries to close the console from a modified state. Clicking the Cancel button will reload the data on the displayed tab and clear the MODIFIED status.



ConnType: ☐ CON ☐ RDP ☐ ICA

Clear Mappings: ✓ ClearDriveMappings ✓ ClearPrinterMappings

Ignore Mappings: ☐ D ☐ E ☐ F ☐ G ☐ H ☐ I ☐ J ✓ K ✓ L ✓ M ☐ N ☐ O ☐ P ☐ Q ☐ R ☐ S ☐ T ☐ U ☐ V ☐ W ☐ X ☐ Y ☐ Z

LpHardMap: ☐ Enabled

Display Settings: ☐ RunSilent ✓ Force Visible Minimum Display Time: 4 Language ID: 0

Flush Token Cache: 14

OU Offsets: User OU Offset: 1 Computer OU Offset: 0

Domain Trusts: Allowed Domains: Denied Domains:

NAC Settings: ✓ Enable NAC Timeout: 30 NAC Command Delay: 5
NAC Command: remediate.bat

Machine Specific: ☐ Enable Machine Specific Config Replace Mode

Helpdesk Message: Contact the helpdesk to report login script errors!

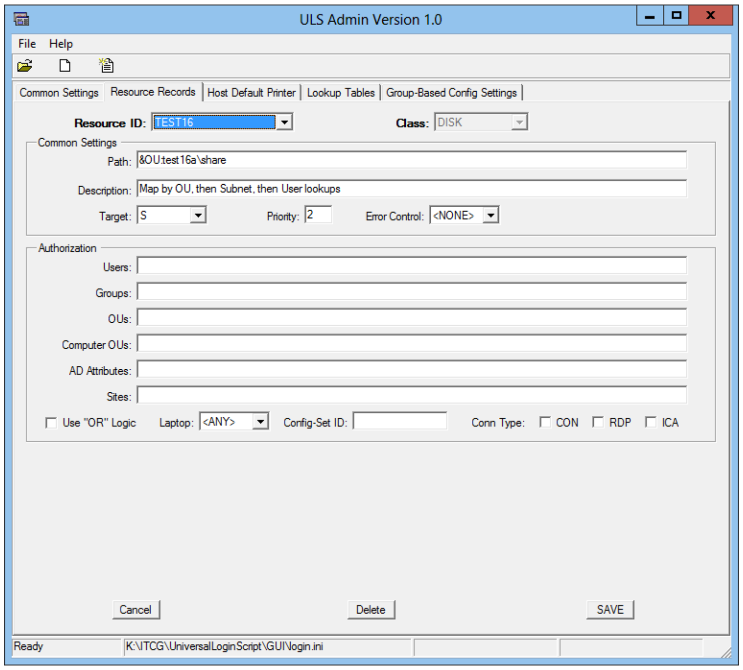
Ready | K:\ITCG\UniversalLoginScript\GUI\login.ini

Several of the controls are interlocked – the Ignore Mappings checkboxes are enabled only when Clear Drive Mappings is enabled; the Display Settings are disabled if Run Silent is not enabled; and the NAC configuration settings are enabled only when NAC support is enabled.

The toolbar has buttons to Open a file; create a New file; and Edit the active file using Notepad. These functions are mirrored on the File menu.

NOTE: Creating a new file will start with an empty config file – Boolean values will be written as displayed, but value fields will be removed if blank. The ULS application employs sane default settings and does not require any parameters defined in the Common section.

NOTE: Opening the active configuration file in Notepad will lock the Admin Console against any input. The status bar will show *** LOCKED *** in the status field and “NOTEPAD ACTIVE” in the Warning field. Input will be ignored, and the Admin Console may report “not responding” while Notepad is running. When Notepad is closed and the file saved, any changes made in Notepad will automatically be reflected in the display.



Resource ID: TEST16
Class: DISK

Common Settings
Path: &OU:test16a\share
Description: Map by OU, then Subnet, then User lookups
Target: S
Priority: 2
Error Control: <NONE>

Authorization
Laptop: <ANY>

Ready
K:\ITCG\UniversalLoginScript\GUI\login.ini

On the Resource Records tab, you can browse any of the records in the config file or create a new record. Selecting a Resource ID will automatically load all of the appropriate data fields, and may display a class-specific group of input fields for Printers, Messages, and Commands. Like the Common tab, certain Common settings are interlocked to the resource Class as Target and Error Control are not used by Message or Command records.

The Class field is unlocked only when New Record is selected. When an existing record is selected, the Class is set to the record's defined class and cannot be changed.

The Cancel button will discard any changes, clear the Modified status, and reload the current resource record's data.

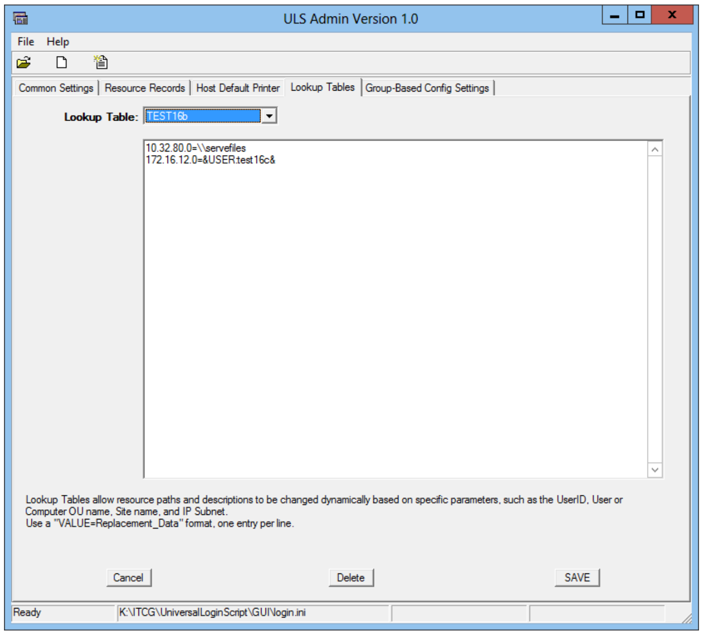
The Delete button will prompt for confirmation of the delete if a valid record is selected, and – if confirmed – will permanently remove the selected Resource ID from the active configuration file. The Delete button is ignored when New_Resource is selected.

The Save button will save the currently defined settings to the configuration file. Note that if you remove the text from a field, the corresponding attribute will be removed from the resource record. A check is performed when Save is clicked to insure that any required fields have data. If these fields are not complete, a warning dialog will display to identify the missing data. Currently, the Class and Path are required for all records, and the Target is required for Disk records. All other fields are optional. If you click Save when New_Resource is active, you will be prompted for the name of a new resource. If you enter an existing resource name, the save will be aborted and a warning displayed in the status bar.

NOTE: The Cancel, Save, and Delete buttons are specific to each tab. If you modify data on one tab, you cannot select a different tab until the changes are saved or canceled.

When the configuration file is first loaded, the Resource Records tab will be blank. You must select a Resource ID from the list to populate the fields. Once a Resource ID is selected, you can scroll through the resource records with the up/down cursor keys or the mouse scroll-wheel.

FUTURE: The small button to the right of each of the Authorization fields will allow the operator to lookup objects for the defined field. Any object selected will be added to the field.



Lookup Table: TEST16b

```
10.32.80.0=\\servefiles
172.16.12.0=&USER:test16c&
```

Lookup Tables allow resource paths and descriptions to be changed dynamically based on specific parameters, such as the UserID, User or Computer OU name, Site name, and IP Subnet.
Use a "VALUE=Replacement_Data" format, one entry per line.

Ready | K:\ITCG\UniversalLoginScript\GUI\login.ini

The remaining tabs (Default Printer, Lookup Tables, & Group Based Config) have a single field that allows all related records to be edited in a text window. The user must maintain proper formatting of these records.

The Lookup Tables tab is show here and has a drop-down field to select one of the detected lookup tables. The other tabs are similar but do not have a selection field as they represent only a single lookup table.

The Save, Delete, and Cancel buttons affect *all* of the data in the edit window. As the editing window is fairly small, we recommend that Notepad be used for editing these sections when more than a few records are defined. If you need to delete a single entry in the table, select the line of text and press the delete key.

Installing the GUI Admin Tool

The Universal Login Script package includes a *Management* folder. Make this folder available to the computer where you wish to install the ULS Admin tool. You may share the Management folder from a central server or copy the folder and its contents to the TEMP folder of the local computer. We recommend launching a command prompt and running the SETUP.BAT file from the command line so that any messages can be reviewed.

No prep or arguments are needed – simply run the SETUP.BAT file. The installer will create the installation in %PROGRAMFILES(X86)%\ITCG, register the Kixforms.dll file, and create a shortcut on the Start Menu in an ITCG Tools folder.

Technical Support

Feedback

We welcome your feedback – comments & suggestions from our users is how the products we produce improve. You can contact us at *Support@innotechcg.com* – please use "Login Script" in the subject line to help route it quickly to the right team. We listen! Recent user suggestions have resulted in improvements such as

- Division-specific config files
- Multi-language support
- Subnet-based Path Rewriting
- User-defined access controls
- Per Computer printer defaults

Support

Having problems making something work the way you want? Drop us a line! Users of our login script can make use of two *free* email-based support instances. We'll help you find a solution to those tough resource allocation challenges, or debug why a resource isn't mapping as you might expect it to.

We also offer commercial support, ranging from setting up your entire configuration file to writing custom enhancements. Fees are reasonable and payable through PayPal and most credit cards. Support is available via email (24x7 with 24-hour response) or by phone (weekdays, 7am-8pm, EST) at 973-272-2667.

See our web site for current support rates.

When you request support, please include the following items in a Zip file attachment:

- A copy of your *login.ini* file.
- A LoginDebug.log file with the result of a login or test process.
  (create an empty %USERPROFILE%\LoginDebug.log and log in).
- The LoginPerf.log file (if enabled) with process timestamps. This should be enabled and included any time that performance is a concern.
- Any error files that result from a login process – see %USERPROFILE%\LoginErr.Log. A capture of any O/S error messages on the screen would also be helpful.

Send your question and Zip file with the above files to *support@innotechcg*.com, with a subject line of "Login Script Support". Although unnecessary, feel free to obfuscate any server names in your configuration and log files.

Appendix 1

Language Locale IDs

These are the ID numbers that represent the various language locale IDs. To support Danish language messages, for example, a file called lsl_2067.lng would be placed into the netlogon share where the login script files are.

1078	Afrikaans	1028	Chinese (Taiwan)
1052	Albanian	1050	Croatian
1118	Amharic (Ethiopia)	4122	Croatian (Bosnia/Herzegovina)
5121	Arabic (Algeria)	1029	Czech
15361	Arabic (Bahrain)	1030	Danish
3073	Arabic (Egypt)	1125	Divehi
2049	Arabic (Iraq)	2067	Dutch (Belgium)
11265	Arabic (Jordan)	1043	Dutch (Netherlands)
13313	Arabic (Kuwait)	1126	Edo
12289	Arabic (Lebanon)	3081	English (Australia)
4097	Arabic (Libya)	10249	English (Belize)
6145	Arabic (Morocco)	4105	English (Canada)
8193	Arabic (Oman)	9225	English (Caribbean)
16385	Arabic (Qatar)	16393	English (India)
1025	Arabic (Saudi Arabia)	6153	English (Ireland)
10241	Arabic (Syria)	8201	English (Jamaica)
7169	Arabic (Tunisia)	5129	English (New Zealand)
14337	Arabic (U.A.E.)	13321	English (Philippines)
9217	Arabic (Yemen)	7177	English (South Africa)
1067	Armenian	11273	English (Trinidad)
1101	Assamese	2057	English (United Kingdom)
2092	Azeri (Cyrillic)	1033	English (United States)
1068	Azeri (Latin)	12297	English (Zimbabwe)
1069	Basque	1061	Estonian
1059	Belarusian	1080	Faroese
2117	Bengali (Bangladesh)	1065	Farsi
1093	Bengali (India)	1124	Filipino
5146	Bosnian (Bosnia/Herzegovina)	1035	Finnish
1026	Bulgarian	2060	French (Belgium)
1109	Burmese	11276	French (Cameroon)
1027	Catalan	3084	French (Canada)
3076	Chinese (Hong Kong S.A.R.)	9228	French (Congo, DRC)
5124	Chinese (Macau S.A.R.)	12300	French (Cote dIvoire)
2052	Chinese (PRC)	1036	French (France)
4100	Chinese (Singapore)	5132	French (Luxembourg)

13324	French (Mali)	1086	Malay (Malaysia)
6156	French (Monaco)	1100	Malayalam
14348	French (Morocco)	1082	Maltese
10252	French (Senegal)	1112	Manipuri
4108	French (Switzerland)	1153	Maori (New Zealand)
7180	French (West Indies)	1102	Marathi
1122	Frisian (Netherlands)	1104	Mongolian (Cyrillic)
1071	FYRO Macedonian	2128	Mongolian (Mongolia)
2108	Gaelic Ireland	1121	Nepali
1084	Gaelic Scotland	0	None
1110	Galician	1044	Norwegian (Bokmal)
1079	Georgian	2068	Norwegian (Nynorsk)
3079	German (Austria)	1096	Oriya
1031	German (Germany)	1045	Polish
5127	German (Liechtenstein)	1046	Portuguese (Brazil)
4103	German (Luxembourg)	2070	Portuguese (Portugal)
2055	German (Switzerland)	1094	Punjabi
1032	Greek	1047	Rhaeto-Romanic
1140	Guarani (Paraguay)	1048	Romanian
1095	Gujarati	2072	Romanian (Moldova)
1037	Hebrew	1049	Russian
1279	HID (Human Interface Device)	2073	Russian (Moldova)
1081	Hindi	1083	Sami Lappish
1038	Hungarian	1103	Sanskrit
1039	Icelandic	3098	Serbian (Cyrillic)
1136	Igbo (Nigeria)	2074	Serbian (Latin)
1057	Indonesian	1072	Sesotho
1040	Italian (Italy)	1113	Sindhi
2064	Italian (Switzerland)	1115	Sinhalese (Sri Lanka)
1041	Japanese	1051	Slovak
1099	Kannada	1060	Slovenian
1120	Kashmiri	1143	Somali
1087	Kazakh	1070	Sorbian
1107	Khmer	11274	Spanish (Argentina)
1111	Konkani	16394	Spanish (Bolivia)
1042	Korean	13322	Spanish (Chile)
1088	Kyrgyz (Cyrillic)	9226	Spanish (Colombia)
1108	Lao	5130	Spanish (Costa Rica)
1142	Latin	7178	Spanish (Dominican Republic)
1062	Latvian	12298	Spanish (Ecuador)
1063	Lithuanian	17418	Spanish (El Salvador)
2110	Malay (Brunei Darussalam)	4106	Spanish (Guatemala)

18442	Spanish (Honduras)
3082	Spanish (International Sort)
2058	Spanish (Mexico)
19466	Spanish (Nicaragua)
6154	Spanish (Panama)
15370	Spanish (Paraguay)
10250	Spanish (Peru)
20490	Spanish (Puerto Rico)
1034	Spanish (Traditional Sort)
14346	Spanish (Uruguay)
8202	Spanish (Venezuela)
1072	Sutu
1089	Swahili
1053	Swedish
2077	Swedish (Finland)
1114	Syriac
1064	Tajik
1097	Tamil
1092	Tatar
1098	Telugu
1054	Thai
1105	Tibetan
1073	Tsonga
1074	Tswana
1055	Turkish
1090	Turkmen
1058	Ukrainian
1056	Urdu
2115	Uzbek (Cyrillic)
1091	Uzbek (Latin)
1075	Venda
1066	Vietnamese
1106	Welsh
1076	Xhosa
1085	Yiddish
1077	Zulu

Appendix 2

Common AD Attributes

The following is a list of common AD Attributes that can be used for the ADATTR parameter and the AD based Value Rewrite features. Note that this list is only an example of the most common (and most commonly populated) attributes and is by no means exhaustive. Any valid attribute name may be used.

Attribute Name	**Description**
streetAddress	Street Address where user is located
L	City (locale) where the user is located
postalCode	ZIP or Postal Code where the user is located
physicalDeliveryOfficeName	Office name
description	Description of account
co, c, or countryCode	Country (code) where the user is located
title	User's job title
department	Business Department where user works
company	Name of company where user works
manager	Name of person the user works for

There are many online resources that can provide a complete list of AD/LDAP attributes, some of which include:

http://www.selfadsi.org/

http://www.rlmueller.net/References/Schema.xls (XLS file download)

While there are hundreds of attributes, only a small number of them make sense to use to authorize resource access or in Value Rewrite operations.