Enumerating InstallAnywhere Variables at Run Time

Robert Dickau
Principal Technical Training Writer
Acresso Software

InstallAnywhere variables are used to store data that are used by many standard actions, and this data can be shared between actions in a running installation. Predefined InstallAnywhere variables store product-specific information (as in PRODUCT_NAME, PRODUCT_VERSION_NUMBER, and PRODUCT_ID); locations of special directories on the target system (using Magic Folders such as USER_INSTALL_DIR and PROGRAMS_DIR); and other system information (as in FREE_DISK_SPACE_BYTES and JAVA_HOME). InstallAnywhere also exposes environment variables and Java system properties though InstallAnywhere variables.

For troubleshooting purposes, it can be useful to get a list of variables used by a running installation. Beginning with InstallAnywhere 2008 Value Pack 1, you can use the **getVariables** method of the **InstallerProxy** class (and other proxy classes) to obtain an Enumeration that contains the names of all current InstallAnywhere variables.

For example, the following custom code action uses the **getVariables** method to obtain the variable names, and then loops over the enumeration to obtain the value of each variable using the **substitute** method.

```
import com.zerog.ia.api.pub.*;
import java.util.*;

public class ShowAllVars extends CustomCodeAction
{
    public void install(InstallerProxy ip) throws InstallException
    {
        System.out.println("InstallAnywhere variables:");

        Enumeration vars = ip.getVariables( );

        while (vars.hasMoreElements( ))
        {
            String var = (String)vars.nextElement( );
            // when using substitute, add dollar signs around variable
            // name to ensure the value is extracted
            System.out.println("\t" + var + "=" +
                ip.substitute("$"+var+"$"));
        }
    }

    // ...omitting uninstall, getInstallStatusMessage, and
    //    getUninstallStatusMessage methods...
}
```

After compiling the code, place the class file in a .zip file or .jar file, after which you can execute the code using an Execute Custom Code action. After building the installer, you can run it in debug mode (holding down the Ctrl key while launching the installer on Windows, for example) to see the list of variables and values displayed in the console or command prompt:

```
DOLLAR=$
CHOSEN_INSTALL_BUNDLE_NUM=2
DESKTOP=C:\Documents and Settings\Rd\Desktop
prop.path.separator=;
lax.version=9.5
lax.nl.env.exact_case.PROCESSOR_ARCHITECTURE=x86
OVERWRITE_IA_CHMOD=false
PROGRAMS_DIR_32=C:\Program Files
prop.sun.boot.library.path=C:\Program Files\Java\jre1.6.0_07\bin
prop.java.vm.specification.version=1.0
USER_HOME=C:\Documents and Settings\Rd
INSTALL_DRIVE_ROOT=C:\
prop.os.version=5.1
prop.file.encoding=Cp1252
CHOSEN_INSTALL_SET=Typical
```

The entries that begin with “lax.nl.env” are system environment variables, and the other entries beginning with “lax” are LaunchAnywhere settings. The entries beginning “prop” are Java system properties, and the rest of the settings are InstallAnywhere variables.

For more information, see the InstallAnywhere help library.