|  |  |
| --- | --- |
| Scripting Best Practices |  |

|  |  |  |
| --- | --- | --- |
|  | | |
| Using Scripts | | |
|  | | |
|  | Using Unix scripts instead of Informatica mappings is generally a bad idea. NBCU policy is to upgrade Informatica in odd number years to be ready to support the Olympics and Elections in even number years. Upgrading and testing Informatica mappings is much easier and safer than upgrading Unix scripts. | |
|  | | |
|  | Unix scripts should be used instead of Informatica mappings only as a last resort. Never assume that Informatica can’t do something. Never assume that a Unix script can be written and tested faster than an Informatica mapping can be created and tested. Use scripts only ***after*** you have ***proven*** that Informatica cannot provide the required functionality. | |
|  | | |
| One Script Across All Environments | | |
|  | | |
|  | If a script is demonstrated to be the only available solution, the same script should work with no changes in all environments, from Development to Production and Disaster Recovery. This typically requires including conditional logic that sets environment variables based on the host environment. | |
|  | | |
|  | To support this, the NBCU Informatica team maintains a number of standardized environment variables that scripts can call. These are documented below   |  |  | | --- | --- | | **How To use Environment Variables** |  |  |  | | --- | |  | | The script /usr/local/bin/get\_edw\_inf\_env can be used by scripts to set a number of environment variables useful with batch Informatica processing. | |  | | The Enterprise Data Warehouse team maintains this script and the variables. | |  | | Each variable begins with EDW\_INF\_ to distinguish it from variables that are defined by applications in their processing. | |  | | Notice how some variables are all lower case. By convention these variables will only have lower case values. | |  | | Variables that follow the typical Unix convention of all upper case can contain values of any case. | |  | | |  |  |  |  | | --- | --- | --- | --- | | **Variable** | **Description** | **Example** | **Comments** | | EDW\_INF\_APPLICATION | The NBCU application | UKDSE | This is actually not set by get\_edw\_inf\_env. Teams that use the default West Coast (WC) Informatica environment may set this at their option.  Teams that do not use the WC environment will be asked to set and export this variable prior to calling get\_edw\_inf\_env to ensure their apps work properly with their Informatica environment. | | EDW\_INF\_APPL\_HOME | the home of application support, such as scripts, but not data | nbcdevwc01 |  | | EDW\_INF\_APPL\_MOUNT | the fully pathed location where $EDW\_INF\_APPL\_HOME can be found | /nbcdevwc01 |  | | EDW\_INF\_BATCH | The batch server used with this environment | inf96batchwcdev.stg-tfayd.com | Notice how the fully qualified DNS alias is used. | | EDW\_INF\_BIN | the location of Informatica binaries | /infdevwc/Informatica9.6.1/server/bin |  | | EDW\_INF\_DATA\_HOME | the home of application data | infdevwc\_data |  | | EDW\_INF\_DATA\_MOUNT | the fully pathed location where $EDW\_INF\_DATA\_HOME can be found | /infdevwc\_data |  | | EDW\_INF\_DNS\_DOMAIN | The DNS domain of this environment | stg-tfayd.com | Notice how this variable does not start with a period. | | EDW\_INF\_DOMAIN | The Informatica domain | INF96\_DOM\_DEV |  | | EDW\_INF\_DOMAINS\_FILE | The location of the domains.infa file | /infdevwc/Informatica9.6.1/domains.infa |  | | edw\_inf\_env | the Informatica environment expressed as all lower case | dev | * int - Integration / Sandbox * dev - Development * qa - QA * stg - Staging * prod - Production * dr - Disaster Recovery | | EDW\_INF\_ENV | the Informatica environment expressed as all upper case | DEV | * INT - Integration / Sandbox * DEV - Development * QA - QA * STG - Staging * PROD - Production * DR - Disaster Recovery | | EDW\_INF\_INT\_SVC | The integration service | INF96\_PROD\_SVC\_WC\_OSP |  | | edw\_inf\_location | the location of the Informatica environment (lower case) | wc | ec for East Coast, wc for West Coast, dse for the dedicated DSE environment. | | EDW\_INF\_LOCATION | the location of the Informatica environment (upper case) | WC | EC for East Coast, WC for West Coast, DSE for the dedicated DSE environment. | | EDW\_INF\_REPOSITORY | the Informatica repository service | INF96\_REPO\_WC\_DEV |  | | EDW\_INF\_TERADATA\_DB | The Teradata database to use with this environment | TDDEV | Notice how this variable does not include the fully qualified domain name of the Teradata database. | | EDW\_INF\_VERSION | represents the version number of Informatica | 96 | Will not include a period, since that can be problematic when used in environment variables. In this example "96" represents Informatica version 9.6. | | |  | | |
|  | | |
| Redirect stderr to the same location as stdout | | |
|  | | |
|  | The default behavior of any Unix process is write error messages to a different stream (to stderr) than non-error messages (to stdout). A common mistake is to forget to redirect stderr to the same location as stdout. If you forget to do this, error messages can be lost, making it much harder to determine why a script did not behave as expected. | |
|  | | |
|  | Here is an example of how to redirect stderr to the same location as stdout: | |
|  | | |
|  | | [infdevusr@potapld00022 tmp]$ ls -la /home/ushbat > /tmp/log\_list 2>&1  [infdevusr@potapld00022 tmp]$ cat /tmp/log\_list  ls: cannot open directory /home/ushbat: Permission denied  [infdevusr@potapld00022 tmp]$ |
|  | | |
| Check Return Codes – All of Them | | |
|  | | |
|  | A common cause of hard to debug script failures is not checking all return codes. Always check return codes, even for simple commands like ls or cat. | |
|  | | |
| Avoid Piped Commands | | |
|  | | |
|  | When commands are piped together, only the return code of the very last command in the pipe can be checked. Better to execute separate commands and check and react to each return code. | |
|  | | |
| Clean Up Temporary Files | | |
|  | | |
|  | If the other recommendations in this document are followed, a large number of temporary files will be created. Be sure to include logic to remove all temporary files when the script ends. That logic should clean up whether the script ends successfully or unsuccessfully. | |
|  | | |
| Clean Up Old Output Log Files | | |
|  | | |
|  | If you are creating output log files as part of your script, be sure your script cleans up older versions of those files. This will prevent your script from failing because the file system has filled up with your old log files. | |
|  | | |