# Application Initiation

BY
JAMES G. HASSLACHER, JR.

# Application Initiation

Topics

- Definition
- Benefits
- Liabilities
- Competition
- Example

## Application Initiation

What is it?

An in-house script that is executed before the actual application ensuring that the correct environment is in place before relinquishing control to the application.

## Application Initiation

Benefits

- High Level
  - Fault Tolerant
  - Simplifies Desktop Settings
  - Single Point of Control
  - Provides Hooks for In-House Enhancements
- Rexx vs Other Languages
  - Easier to learn and more robust that *.bat
  - Windows executable
  - SAA extensions access controls usually limited to C++

## Benefits

High Level

- Fault Tolerant
  - Regains control of the desktop.  Can search for application or dependencies when user has moved them from the expected location.
  - Can detect and react to failures, unlike MS utilities such as “regedit /s”.
  - Does not rely on static settings (autoexec.bat, *.ini files, or Registry) that are accessible to the user. The initiator can dynamically adjust all settings, including the PATH environment variable.

## Benefits

High Level

- Simplifies Desktop Settings
  - No start-up directory needed in shortcut.
  - Eliminates need for special scripts using WINSET (or other utilities that alter the environment) as a separate command prior to executing the application.
  - Server URL’s can be removed from PATH.
    - A performance side benefit is noticed when network server directories are not searched for commands.

## Benefits

High Level

- Single Point of Control
  - Rexx interpreter, scripts, and static control files stored in a server directory enables one change to affect all desktops.
  - Server directory can be locked down to prevent user meddling.

## Benefits

High Level

- Provides Hooks for In-House Enhancements
  - “Self Healing”, or self updating features for the application can be provided by the initiator.
  - Network can be tested, and the user given the option to bail if speed or other resources are not available.

## Benefits

Rexx vs Other Languages

- Easier to learn and more robust than *.bat
  - Self-evident
  - DOS executable
    - Different types of shortcut's are loaded on an NT, than 9x.  NT requires a *.PIF for DOS, and a *.lnk for Win32 executables.  9x uses *.lnk for both.
    - Requires environment size to be defined for scripts that use the SET a lot.  If left unset, in the auto state, it will cause the script to fail.

## Benefits

Rexx vs Other Languages

- Windows Executable
  - One icon does all
    - DOS executable
      - Windows 9x uses a *.lnk file
      - Windows NT requires a *.pif
  - Will not exhaust environment space (DOS concept)
  - Can execute without user awareness.  A window or console is not required, unless interaction with the user is necessary.

## Benefits

Rexx vs Other Languages

- SAA extensions access controls usually limited to C++
  - Access to Registry
  - Database Queries
  - Executable’s Properties

## Application Initiation

Liabilities

- Interpreted Language
- Single Point of Failure

## Liabilities

Interpreted Language

- Slow
  - Tradition holds that Interpreted languages are slower than compiled ones, due to the fact that each line must be “re-compiled” each time it is executed.  With the speed of today’s PC processors this difference is only detectable at the hardware level.
- Resource intensive
  - Again tradition is trashed.  Regina 0.08h (text interpreter) vs WinWord 8.0 (graphics app) is 300KB (total) vs 5MB (executable without *.dll’s.)

## Liabilities

Single Point of Failure

- If the files server with the Application Initiator cannot be reached (crashed, network down, etc.), then none of the apps that it starts can be used.
  - All of the development teams that have used this approach have stated that if the environment cannot be set exactly as specified, this is the desired behavior.
  - An alternative method for initiating apps locally has been developed, but not requested.

## Application Initiation

Competition

- Microsoft Scripting Languages
- TNSNAMES Server
- Windows 2000

## Competition

Microsoft Scripting Languages

- *.bat
  - Discussed in slide 10.
- Windows Scripting Language
  - Based on Jscript (MS’s version of JavaScript) and touted as the directionof the future, however, it is not extensible like Rexx is.
- Visual Basic
  - If IBM were to persue this aggressively with Object Rexx, VB could be wiped off the face of Marketing Sheets everywhere.

## Competition

TNSNAMES Server

- TNSNAMES Server is a product provided by ORACLE to resolve database names into server names, IP addresses, and IP ports. This moves those parameters from the desktop to a server for single point of control.
  - The IT department does not control the desktop. If the user re-installs ORACLE client, then the SQL.ORA file (by default) no longer points at the Names Server, but points to an unitialized TNSNAMES.ORA file, instead.

## Competition

Windows 2000

- The latest release of Windows from Microsoft. It attempts to eliminate the need for Application Initiators by providing utilities such as: self-healing systems, Intellimirror, and ASD.
  - It requires both user and server run Windows 2000.
  - It does not log “self-healing” events, failures, or any other activity.
  - Only reacts to file changes, it does not understand non-file objects. It will not enforce GUID consistency across the enterprise.

## Application Initiation

Example

This example is divided into two parts. The first part is general and can be used by many applications. The second part sets the environment specific to the application being launched.

- General Initiation
  - ORACLE Setup
- Specific Application Setup
  - PeopleSoft 6.0 is the sample application

## Application Initiation

Example

The command syntax is:

Init_Ora_App [product name]

In the example this is:

Init_Ora_App PeopleSoft

## Application Initiation

Example

The following files are used:

| | |
|---|---|
| Init_Ora_App.exe | Initiator |
| Ora.setup | Main script |
| Product.setup | Substitution variables |
| Domain-help.tbl | Telephone numbers |
| PeopleSoft.setup | Main product script |
| Product-domain.tbl | Substitution variables |
| PeopleSoft.psreg | Registry settings |
| PeopleSoft.psalwaysreg | Substitutable Registry |
| PeopleSoft.environment | Substitutable env vars |
| PeopleSoft.PATH | Substitutable PATH vars |

## Example

General Initiation

- Find the ORACLE executables
- Find the Admin directory
- Update the TNSNAMES.ORA file
- Update the PATH
- Make the ORACLE executable directory the current directory.
- Call the script specific for the application.

## Example

Specific Application Setup

- Create the correct Registry entries
- Setup the environment variables
- Add the necessary directories to the PATH
- Launch the application

## General Initiation

Find the ORACLE executables

- Examine the data in the Value pointed to by the Registry Key - Attribute named "HKEY_LocalMachine\SOFTWARE\ORACLE\ORACLE_HOME"
- Append "\bin" to that Value.

## General Initiation

Find the Admin directory

- Get the version to the file OCIW32.DLL, the ODBC driver.
  - X is version 7 of the ORACLE client; Y is version 8 of the client.
- Find the correct NETxx Attribute in the ORACLE Registry Key.
  - NET20 for version 7
  - NET80 for version 8
- The Value of the NETxx Attribute, plus "\Admin", is the path.

## General Initiation

Update the TNSNAMES.ORA file

- If the SQLNET.ORA file points to TNSNAMES, not the Names Server, then:
  - Look for an old entry (listed by database name) in the TNSNAMES.ORA
  - Remove it, if it exists.
  - Add an entry using the information in the product.setup file.

## Specific Application Setup

Create the correct Registry entries

- Read the Registry entries from the PeopleSoft.psalwaysreg file.
  - Perform any necessary variable substitution
- Issue an error message if one of the variables cannot be set.
  - Give the user the option to continue or abort the installation.

## Specific Application Setup

Setup the environment variables

- Read the variables and values from PeopleSoft.environment file.
  - Perform any necessary variable substitution
- Set them using the built-in Rexx function. They will only persist as long as this process, or its children, exists.
  - Prevents filling the environment space for applications other than PeopleSoft.

## Specific Application Setup

Add the necessary directories to the PATH

- Extract the list of path’s to append from the PeopleSoft.PATH file.
- Convert all path’s in the PATH variable from short names to long names
- Append the new paths to the PATH
- Remove duplicates