



Product Bulletin

Bulletin Number: P-2007-0218-Global
Date: 20 August 2007

New CallPilot Support Utility - AppBuilder Explorer

REVISION HISTORY

Date	Revision #	Summary of Changes
20 August 2007	Original bulletin	This is the original publication

Introduction

The bulletin introduces the new CallPilot support utility, *Application Builder Explorer*, for use in identifying potential problems with the design and call-flow routing of CallPilot applications such as auto-attendant menus and announcement services.

Application Builder Explorer Utility Details

The CallPilot Application Builder Explorer (ABExplorer) support utility is yet another in the comprehensive suite of supplemental maintenance capabilities for CallPilot. This utility minimizes the amount of time required to identify potential problems with the design and call-flow routing of CallPilot applications, specifically identifying applications containing loops. Some infinite loops, which if encountered, may negatively impact the system operation, or at a minimum, be a nuisance to callers resulting in "voice-mail jail" type scenarios.

This document also supplements the NTPs and online help, providing guidance on proper application development and additional information specific to the use of various loop types within a call flow, outlines the potential impacts of infinite loops, and provides guidance on the use of the utility.

This diagnostic utility is non-obtrusive and can be safely run on a client PC with connectivity to CallPilot servers running releases 2.02, 2.5, 3.0, 4.0, and 5.0. For complete details, reference [Appendix-A](#).

Ordering Guidelines and Procedures

No changes to ordering or provisioning occur due to the introduction of this support utility. It is available for download at no additional cost.

References and Related Documents

For additional information on the use of Application Builder, reference the following NTPs:

CallPilot 5.0 – NN44200-102 Application Builder Guide.

CallPilot 4.0 and earlier – NTP 555-7101-325 Application Builder Guide

These NTP documents are available for download from the Nortel Technical Support Portal (TSPortal) website at www.nortel.com/support under CallPilot / Documentation

Obtaining the Utility

Application Builder Explorer Utility (ABExplorer) is available for download from the Nortel Enterprise Solutions PEP Library (ESPL) and Technical Support Portal (TSPortal) websites.

ESPL: www.nortel.com/espl

Use PEP ID “ABExplorer”

TSPortal: www.nortel.com/support

Under Product-CallPilot / Support Tools tab

Utility usage

For complete details on the use of the Application Builder Explorer Utility (ABExplorer), reference [Appendix-A](#) of this document.

*Nortel, the Nortel logo and the Globemark are trademarks of Nortel.

©2007 Nortel Networks Limited. All rights reserved. Nortel, the Nortel logo, and the Globemark design are trademarks of Nortel Networks Limited. All other trademarks are the property of their respective owners.

The information in this document is subject to change without notice. Nortel reserves the right to make changes, without notice, in equipment design as engineering or manufacturing methods may warrant. The statements, configurations, technical data, and recommendations in this document are believed to be accurate and reliable, but are presented without express or implied warranty. Users must take full responsibility for their applications of any products specified in this document. The information in this document is proprietary to Nortel Networks Limited.

To view the most recent version of this bulletin, please visit Nortel's Partner Information Center on the web at: <http://www.nortel.com/pic>.

Appendix-A

Introduction

Application Builder Explorer (ABExplorer) is a utility that enables the CallPilot Application support person to quickly identify potential service affecting loops within their AppBuilder applications.

This document describes:

1. [Loops](#)
2. [Loop Types](#)
3. [Potential Impacts of Loops](#)
4. [Using the ABExplorer Utility](#)

1. Loops and Loop Types

Loops are a useful mechanism to provide intelligence to the interactive applications produced with CallPilot Application Builder (AppBuilder). Depending on input (either from user or system), application call flow can be looped-back to a previously used block (section) where appropriate.

A loop is a subsection of the design in which the output(s) from the subsection connect to the input(s) of that same subsection. The simplest AppBuilder example of a loop would be a single block which connects one of its outputs to one of its inputs.

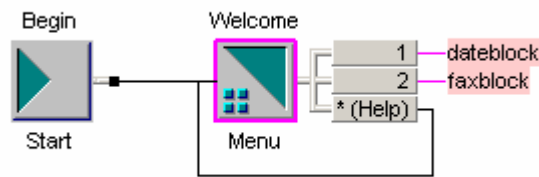
In more generic terms the subsection would consist of a series of AppBuilder blocks tied together. An output from this subsection would get looped back to an input of this subsection.

A particular loop may get executed several times during a call, depending on the loop design. A loop has a body and set of boundary conditions. The loop body is executed until the boundary condition is true.

When using this method of designing an AppBuilder application, the designer must follow certain precautions.

The following is an example of a simple loop:

1. Provides "Welcome" announcement to explain options to caller.
2. If caller requires "Welcome" message be repeated, returns to step 1 (i.e. if caller presses asterisk '*' key), otherwise proceed to step 3 (i.e. caller presses '1' or '2').
3. Proceed to 'dateblock' or 'faxblock', according to choice of caller.



Note: In this example, the loop body is executed until the user selects one of the two boundary conditions, dateblock (presses “1”) or faxblock (presses “2”).

2. Loop Types

Loops can be divided into several categories according to their boundary conditions:

1. Contains a hard-coded number of iterations. In AppBuilder, such loops are blocked from being created. However, they can be propagated by using the “Copy & Paste” command.
2. Contains an undefined number of iterations. In such loops, the boundary condition can be false under some conditions. This category can be further subdivided:
 - a. Boundary condition depends on user input.
 - b. Boundary condition independent of user input.
3. Contains an infinite loop. Boundary condition is always true preventing the application from exiting the loop.

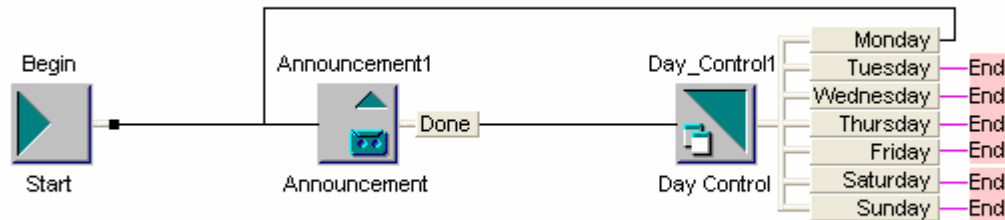
Close attention needs to be placed on the boundary condition when implementing a loop.

The following is an example of a simple “**Infinite**” loop where the application never exits.

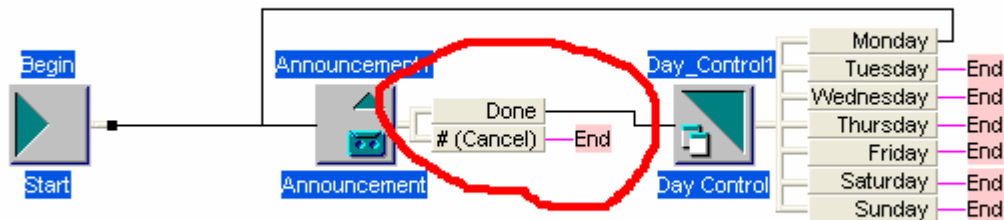


Note: Applications containing infinite loops will be terminated or will never exit. It depends on block type that is involved with the loop.

The following is an example poor implementation of loops. While it offers mostly, **“Non-Infinite”** loops (because each has an exit); it acts like an **“Infinite”** loop whenever the day-of-week is Monday.



A better implementation of the above application is shown below. The application was modified by providing an interruption in the loop. The caller can press the octothorpe or pound “#” key to exit the loop (and application) and avoid abnormal termination.



Important Rule:

All loops require an exit. Ensuring that all paths through a loop have a way out will protect your application from consuming excessive CPU resources and degrading system performance.

3. Potential Impacts of Loops

AppBuilder Application designers must be aware of loops in their design and use these loops responsibly. In this section, we are going to talk about the impacts of loops on the system.

There are two possible scenarios that can result from encountering an infinite loop:

1. **Simple Loop:** Abnormal termination of the application. If the loop is simple (contains only one block), SLEE usually can detect that the application entered an infinite loop and appropriately terminate it.

This is not a recommended condition and under certain scenarios could impact the overall system performance.

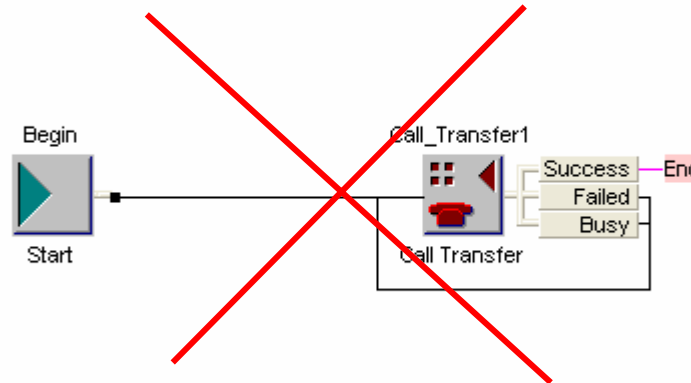
2. **Complex Loop:** SLEE can not detect that the application entered an infinite loop and does not attempt to terminate it. Applications of this type can stay trapped in an infinite loop until the next system reboot leading to excessive CPU usage by the SLEE service. CallPilot resources assigned to this call will remain busy and in use long after the originator has left.

This condition will result in some negative effects:

- CallPilot will start to answer slowly or not at all. Allocated resources never get released. CallPilot eventually runs out of resources (DSPs, SLEE channels, etc.) resulting in a Ring-No-Answer (RNA) condition.
- The caller that entered the infinite loop will most likely have not obtained the intended service.

Special attention needs to be paid to loops involving “Call Transfer” and “Thru-Dial” blocks. “Call Transfer” and “Thru-Dial” blocks that have “failed” or “busy” outputs that are redirected back into their respective inputs can result in serious call processing issues such as “All Ports Busy” (Ring-No-Answer), and Phantom Calls.

The following example shows a configuration that is prone to causing an infinite loop. When the “Failed” or “Busy” conditions exist, the block will infinitely loop, continuously attempting to successfully transfer the call. While in this “Failed” or “Busy” condition, excessive system resources can be consumed in trying to establish the transfer. The loop does not depend on user input and thus it will be repeated as fast as system resources allow. **This excessive use of resources has been shown to result in system failure. The use of such loops should generally be avoided.**



An improvement to this design would be to add a prompt in the loop path that would require user input. Much less system resources are used since now the system has to wait for a user to press a key. In addition, it would only be an infinite loop for as long as the user decided to remain providing input.

The issue could be completely avoided by changing the design to not contain a loop. For instance a “Failed” or “Busy” condition could be routed to an attendant.

4. Using the ABExplorer Utility

ABExplorer is a diagnostic tool to allow Nortel and Channel Partner technical support personnel to perform basic analysis on AppBuilder applications. The utility scans the entire suite of AppBuilder applications for loops by parsing the source files (.ED) for the following information:

1. Blocks (e.g. Call Transfer, Thru-dial, etc.) that loop back onto themselves.
All AppBuilder block types can be configured as search parameters.
2. Version of the AppBuilder Editor and SNT compiler that was used to create an application.
3. Voice and Fax Items within the applications.

Note: The utility is intended for use by qualified/trained technical support teams.

Using ABExplorer

Due to high amounts of disk activity and possible contention between ABExplorer and the AppBuilder Editor, this utility can not be executed on a CallPilot server. Instead, it must be executed from a client PC. The utility contains guardrails to prevent running it on the server. If attempted, a dialog box appears with the following message "Cannot run ABExplorer on a CallPilot server Quitting."

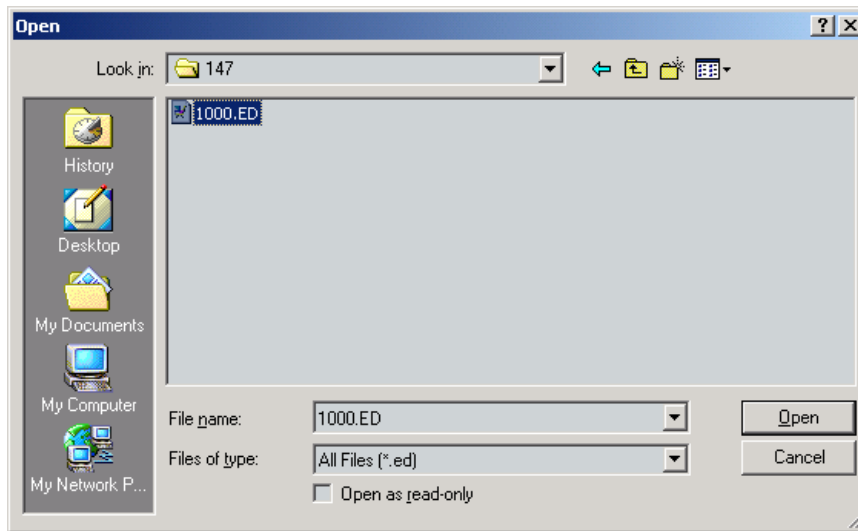
To start analysis in ABExplore:

1. Copy ABExplorer to a folder on a non-CallPilot workstation (Windows platform).
2. Copy the AppBuilder applications from the CallPilot server to a workstation folder named "uprog" where the ABExplorer utility is installed. AppBuilder applications are stored on the CallPilot server in the following locations (larger systems may contain multiple "uprog" folders, located in D:\, E:\, and F:\):
 - D:\Nortel\cust\cust1\nm_abd\uprog\
 - E:\Nortel\cust\cust1\nm_abd\uprog\ (for multi-volume servers)
 - F:\Nortel\cust\cust1\nm_abd\uprog\ (for multi-volume servers)

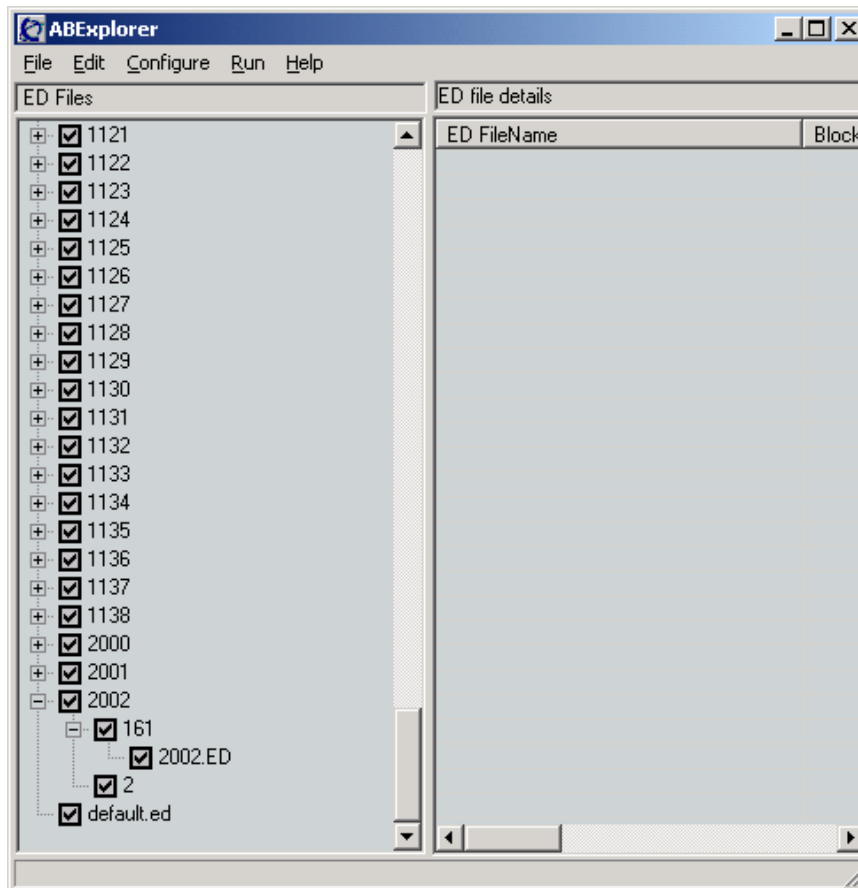
Example of copying applications to a local folder D:\Temp\uprog (path layout):

- D:\Temp\uprog\1000\1\1000.ED
- D:\Temp\uprog\1000\1\1000.RUN
- D:\Temp\uprog\1000\2\1000.ED
- D:\Temp\uprog\1000\2\1000.RUN
- D:\Temp\uprog\1001\147\1001.ED
- D:\Temp\uprog\1001\147\1001.RUN

3. Launch ABExplorer and select **File > Open**. Use the File/Open dialog to locate a single .ED file within the AppBuilder applications and click **OK**.



4. ABExplorer will locate the uprog folder as the starting location for including applications. The Application IDs (AppIDs) will appear in the left window pane. Select specific AppIDs for analysis.



5. From the **Configure** Menu option the following analysis options are available:



The above dialog box is used to select which AppBuilder application blocks are to be included in the loop search. Click the “**AppBuilder Defaults**” button to set to default configuration. This will search all blocks for loops except for “Announcement” and “Menu” blocks which can naturally be looped back onto their selves.

Explanation of checkboxes in ABExplorer options dialog is listed below.

All:

This checkbox will toggle all or none of the AppBuilder blocks.

Obsolete Announcement:

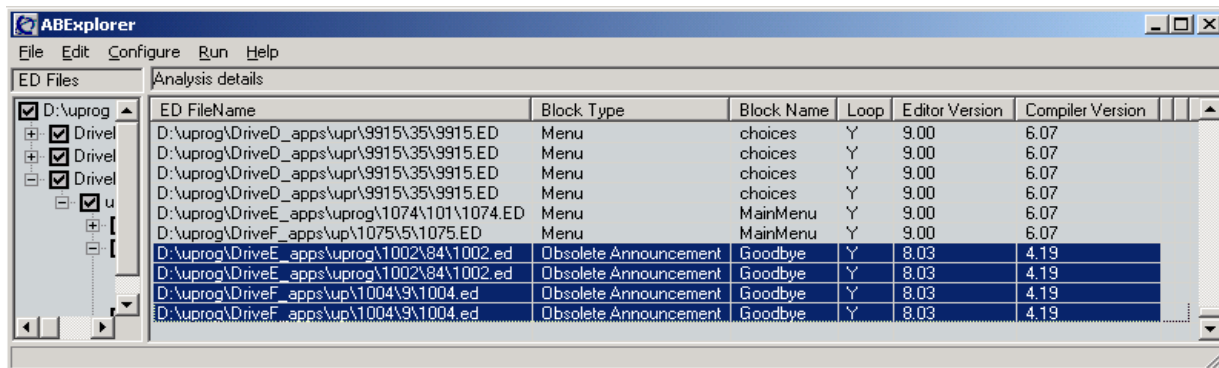
This is a special block that can be searched for. There are no known issues with having this block type. To simplify trouble-shooting efforts, applications with this block type should be opened and saved with the AppBuilder editor to bring the block up to the latest version. To ensure that this change is saved, it may be necessary to move a block in the design (and then move it back to its original position) to signal to AppBuilder that a change has been made.

Version Info:

ABExplorer displays the version number for the AppBuilder editor GUI and underlying compiler used to create the application. These version numbers are used by CallPilot designers to help to determine whether a change to the AppBuilder block interface is associated with any customer issue. As of CallPilot 2.02/Service Update 4 (SU04), the editor version is 9.01 and the compiler version is 8.03. There are no reported issues with having older AppBuilder application versions, but this feature is included for trouble-shooting purposes.

Example:

The highlighted lines in the following ABExplorer screenshot have detected application 1002 and 1004 using an old announcement block interface. Likewise, the version information used to create these applications is 8.03 for the AB Editor and 4.19 for the SNT compiler. It's advisable to update these older applications the current versions by opening, and then saving the applications in AppBuilder.



ED FileName	Block Type	Block Name	Loop	Editor Version	Compiler Version
D:\uprog\DriveD_apps\upr\9915\35\9915.ED	Menu	choices	Y	9.00	6.07
D:\uprog\DriveD_apps\upr\9915\35\9915.ED	Menu	choices	Y	9.00	6.07
D:\uprog\DriveD_apps\upr\9915\35\9915.ED	Menu	choices	Y	9.00	6.07
D:\uprog\DriveD_apps\upr\9915\35\9915.ED	Menu	choices	Y	9.00	6.07
D:\uprog\DriveE_apps\uprog\1074\101\1074.ED	Menu	MainMenu	Y	9.00	6.07
D:\uprog\DriveF_apps\up\1075\5\1075.ED	Menu	MainMenu	Y	9.00	6.07
D:\uprog\DriveE_apps\uprog\1002\84\1002.ed	Obsolete Announcement	Goodbye	Y	8.03	4.19
D:\uprog\DriveE_apps\uprog\1002\84\1002.ed	Obsolete Announcement	Goodbye	Y	8.03	4.19
D:\uprog\DriveF_apps\up\1004\9\1004.ed	Obsolete Announcement	Goodbye	Y	8.03	4.19
D:\uprog\DriveF_apps\up\1004\9\1004.ed	Obsolete Announcement	Goodbye	Y	8.03	4.19

ABExplorer has detected obsolete announcement blocks.

Background:

SNT (Show-N-Tel): The IVR application creator. It consists of an editor and underlying functional blocks (i.e. Call Transfer, Menu, Password Check, etc.). SNT is used by both the Nortel CallPilot Telset Applications designers to create "canned applications" (Voice Messaging, Speech-Activated Messaging (SAM), etc.) and CallPilot administrators (Channel Partners and End-users/Customers, to create customer AppBuilder applications).

When the SNT editor or underlying functional blocks are changed, incompatibilities may occur. There are two versions used to track these differences:

1. AB File Version or Editor Version.
2. SNT Version.

AB file version (#1) is to track AppBuilder application version and SNT version (#2) is used to track the canned apps file versions.

The following is a cross reference list of the CallPilot server software release version, AppBuilder Editor version, and SNT version.

CallPilot version	AB Editor	SNT version
1.00	9.0	5.06
1.06	9.0	5.06
1.07	9.0	6.07
2.01.26/2.01.27	9.01	8.03
2.5	9.01	8.03
3.0	9.01	8.03
4.0	9.01	8.03
5.0	9.01	8.05

Voice Items:

This option lists all voice items defined within an application. This feature is useful for tracing the same prompt defined in separate applications, and also provides a way to gauge voice item usage.

Fax Items:

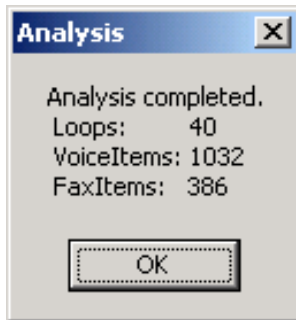
As with voice items, this option lists all fax items defined within an application. This feature is useful for tracing the same fax defined in separate applications. It will also help gauge fax item usage.

- After configuring ABExplorer, the analysis can be started by selecting Menu option **Run->Start Analysis**

Analysis details are displayed within the listview pane in ABExplorer. The following information is listed:

Heading	Description
EDFileName	Used to determine AppID and version information
BlockType	Type of AppBuilder block
BlockName	Name of Block within AppBuilder
Loop	True (Y) if loop found on a particular Block
EditorVersion	Version of AppBuilder editor
CompilerVersion	Compiler version of SNT compiler
Voice Item Name	Name of Voice item within AppBuilder
Fax Item Name	Name of Fax item within AppBuilder

7. An analysis dialog box will eventually appear with the results.



Supplemental Notes:

- A text version of the results can be copied onto the Windows Clipboard for further analysis by using the **Edit > Copy** menu.
- Additional information concerning detected loops, possible negative effects, and recommended solutions can be obtained by double-clicking on the corresponded block.

For example, for looped Call Transfer block, the following information is displayed:

