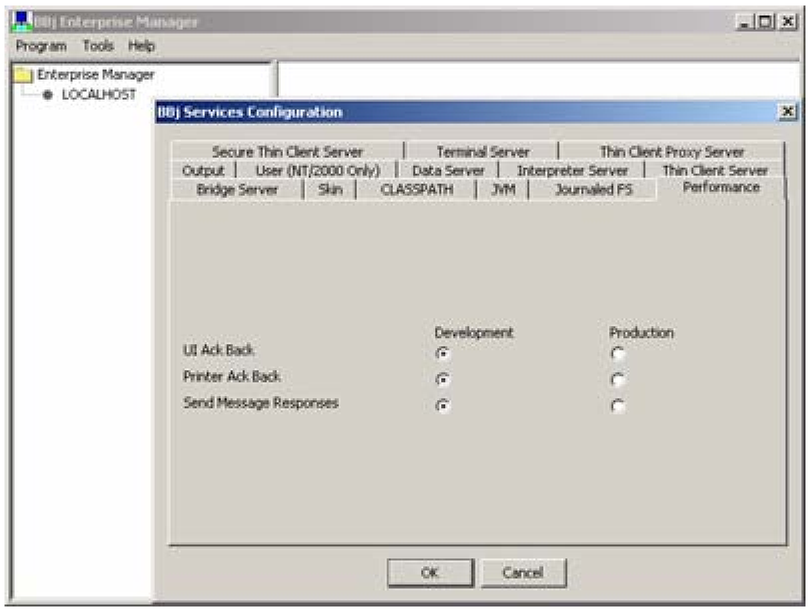
Thin Client Performance Tuning

By Jim Douglas

We first discussed the subject of thin client performance tuning in "The Lessons of BASIS b-Commerce™. Now that we have launched BBj® 2.0, it's time to revisit this subject. BBj 2.0 includes three new options that can be used to reduce the number of messages exchanged between the client and server. These options can be found on the Enterprise Manager "Performance" tab.

Note: These options are only significant in a distributed (thin client) environment.

Enterprise Manager



	Development	Production
UI Ack Back	✓	
Printer Ack Back	✓	
Send Message Responses	✓	

Each time the interpreter issues a command to a remote client, it waits for an acknowledgement from the client before proceeding. This is to ensure that any errors are handled at the correct point in the program. With certain commands, acknowledgements can be safely disabled if an application has been fully tested and if there is no possibility of an error occurring. Acknowledgements can be selectively disabled for three categories of commands:

1. UI Ack Back (User Interface Commands). These include PRINT commands to channel 0 or to a SYSGUI channel.
2. Printer Ack Back (Printer Commands). These include PRINT commands to a printer channel.
3. Send Messages Responses. These include selected SENDMSG() functions for which the response string is typically ignored. See the documentation for BBjConfig::requireAllSendMsgResponses for a complete list of SENDMSG() functions that are affected by this setting.

Parameter	Examples

UI Ack Back	`PRINT 'CS'`
	`PRINT (SYSGUI)'CONTEXT'(0)`

Printer Ack Back	`REM ' NEW PAGE`
	`PRINT (7)'FF'`

Send Message Responses	`REM ' SET GRID CELL TEXT`
	`ROW$=SENDMSG(SYSGUI,GRID,48,ROW,$$,CONTEXT)`
	`CELL$=SENDMSG(SYSGUI,GRID,22,COL,TEXT$,CONTEXT)`

These settings can also be controlled at runtime by using the following method calls. These method calls can be used to temporarily enable acknowledgements for a block of code that depends on correct error handling:

```
REM ' Development Mode (default)
bbjapi().getConfig().suppressUIAckBack(0)
bbjapi().getConfig().suppressPrinterAckBack(0)
bbjapi().getConfig().requireAllSendMsgResponses(1)

REM ' Production Mode
bbjapi().getConfig().suppressUIAckBack(1)
bbjapi().getConfig().suppressPrinterAckBack(1)
bbjapi().getConfig().requireAllSendMsgResponses(0)
```

See the BBj documentation for more details.

Release On Lost Connection

BBj 2.01 adds a new option for controlling what happens to the server side of a thin client process if the connection to the client side is unexpectedly lost. The default is for the interpreter to RELEASE the server side of the session as soon as the client connection is lost. This behavior can also be selected at runtime with:

```
REM ' A dropped client connection will release the program
bbjapi().getConfig().releaseOnLostConnection(1)
```

The application programmer can gain more control over the process with code like this:

```
REM ' A dropped client connection will trigger a SETESC interrupt
bbjapi().getConfig().releaseOnLostConnection(0)
SETESC escape_handler
:
escape_handler:
IF and(chr(tcb(19)),$08$)=$08$ THEN GOSUB cleanup; release
```

A dropped client connection causes an interrupt that is trapped by the SETESC. In the SETESC handler, the program tests to see if the interrupt was triggered by a dropped client connection. The typical action would be to perform any last-minute cleanup, then release the process. The program must not attempt to communicate with the client at this point. Since the client no longer exists, this can trigger an infinite error/retry loop that can only be interrupted by killing the process in Enterprise Manager.

Summary

In "The Lessons of BASIS b-Commerce™" we learned how important it is to keep bandwidth and latency issues in mind when developing a thin client application. BBj 2.0 provides the developer with even more ways to reduce latency delays. These principles will help you to make your thin client application as efficient and responsive as your customers expect.