Chapter 18.

June 2008

Welcome to the June 2008 edition of IBM Information Server Developer's Notebook. This month we answer the question;

> How do I make use of some of the extensibility of IBM Information Server; more specifically, How do I link in C/C++, Java, and other type program code into my DataStage component Jobs?
>
> *Excellent question! Besides every built in capability that IBM Information Server has, you can also extend its functionality using various APIs and standard programming languages. The month's author prefers Java, but many people prefer C/C++; in part your preference may depend on what previously created program assets you are trying to preserve.*

In short, we are going discuss all of the relevant terms and technologies above, and provide examples that detail how to deliver this functionality using IBM Information Server.

Software versions

All of these solutions were *developed and tested* on IBM Information Server (IIS) version 8.01, FixPak 1A, using the Microsoft Windows XP/SP2 platform to support IIS client programs, and a RedHat Linux Advanced Server 4 (RHEL 4) FixPak U6 32 bit SMP server (Linux kernel version 2.6.9-67.EL-smp) to support the IIS server components.

IBM Information Server allows for a single, consistent, and accurate view of data across the full width of the corporate enterprise, be it relational or non-relational, staged or live data. As a reminder, the IBM Information Server product contains the following major components;

> WebSphere Business Glossary Anywhere™, WebSphere Information Analyzer™, WebSphere Information Services Director™, WebSphere DataStage™, WebSphere QualityStage™, WebSphere Metadata Server and Metabridges™, WebSphere Metadata Workbench™, WebSphere Federation Server™, Classic Federation™, Event Publisher™, Replication Server™, Rational Data Architect™, DataMirror Transformation Server™, and others.

Obviously, IBM Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities.

18.1 Terms and core concepts

IIS taxonomy of all (most) things extensible

As stated above, IBM Information Server (IIS) is a large and capable software product, comprised of many standard and optional components. The standard (always present) components to IIS include the parallel framework, the shared metadata repository, and others. The optional components to IIS include DataStage, Information Analyzer, Business Glossary Anywhere, and many others. Some of the IIS optional components are used more by Business Analysts, some more by IT staff and programmers. Some of the IIS optional components naturally call for extensibility, and some do not. Perhaps the most extensible optional component to IBM Information Server is DataStage, and it is this component, and its extensibility, which is the focus of this document.

Note: Wikipedia.com defines extensibility as,

> *In (Information Technology) systems architecture, extensibility means the system is designed to include hooks and mechanisms for expanding/enhancing the system with new capabilities without having to make major changes to the system infrastructure.*

As an example, if you had previously created a really hot piece of C/C++ or Java code, we can make the functionality appear and execute seamlessly on the DataStage Parallel canvas; we could plug in new data processing algorithms, connectivity to new and unforeseen data sources, the list is endless.

Some of the languages and/or linkages that IBM Information Server (IIS) and the DataStage component support for extensibility include;

- C/C++ programming language
- Java programming language
- Web Services invocation
- BASIC programming language
- Operating System (OS) Utilities; allowing access to Perl, Awk, Sed, and dozens more.
- XML Style Sheet Language Translation (XSLT)

The June/2007 edition of this document (IBM Information Server Developer's Notebook: IISDN) detailed how to use XML and XSLT transforms. XSLT is so powerful, that there are entire hardware/software products in existence that use only XSLT for their data transformation needs. The July/2007 edition of IISDN

detailed Web service invocation and service provision. In the March/2008 edition, IISDN detailed OS Utilities and related.

This document details how to incorporate C/C++, and to a lesser extent, BASIC language programming, into IIS and DataStage Jobs. We are going to table Java topics for this edition of this document at least, preferring to address Java along with possibly Java/JMS and message brokering at some future time.

DataStage Parallel Routines, using C/C++

Just as IBM Information Server (IIS) has an approved (supported) operating system and hardware system platform list, so too there is a list of supported C/C++ compilers and versions. Using whatever C/C++ development workbench you prefer (we prefer Vi and g++), you create and compile a unit of C/C++ code in a particular manner. Using the Designer program to the IIS DataStage component, you may then invoke that C/C++ code through creation and use of a DataStage Parallel Routine.

Figure18-1 displays the first example IIS DataStage Job we will create as an example.

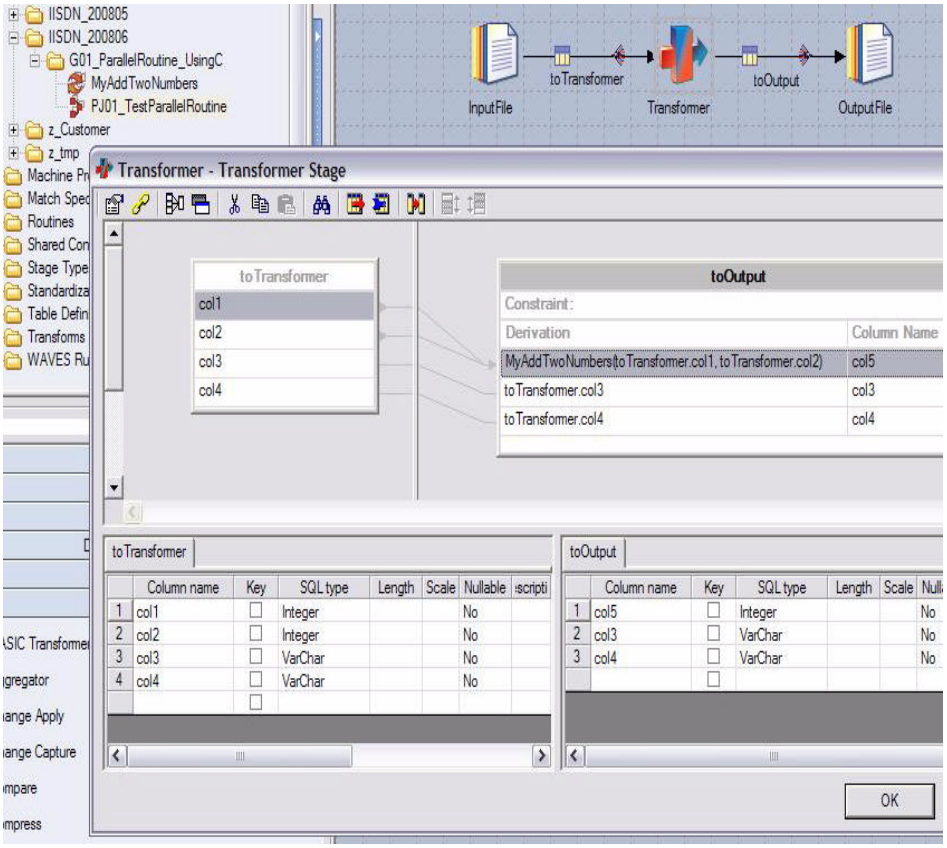

Figure 18-1 First example, using C/C++ in Transformer, with Parallel routine.

A code review related to Figure18-1 follows;

- Figure 18-1 displays the simplest of IIS DataStage component Enterprise Edition Jobs; a simple input (in this case, a Sequential File stage) to a (Parallel) Transformer, to a simple output (another Sequential File stage).
- While the Transformer itself supports endless built in functions and combination of expressions, Figure18-1 displays that the Transformer is invoking a (Parallel Routine) named, MyAddTwoNumbers.

  MyAddTwoNumbers is a piece of custom program code written and compiled in C/C++. One IIS DataStage component technology allowing access to C/C++ code is the Parallel Routine, which we will create later in this document.

IIS extensibility through (new) parallel stage types

In Figure18-1, we are able to invoke a C/C++ language routine mid-stream; that is, in the middle of a DataStage Job input data stream. You may also invoke C/C++ code as the input or output (device) to a DataStage Job. Comments here include;

- There are three extended (home grown, you supply the algorithm and source code) DataStage Parallel Stage types that you may create. These include;
  - Custom Stages
  - Build Stages
  - Wrapped Stages
- Custom Stages are written/configured using a language other than C/C++, Java, etcetera. Custom Stages are written/configured using DataStage OSH Script.

  If you are fluent in OSH Script, you can create a new Parallel Stage in 5 or 10 lines of OSH Script that might take 100-200 lines of code in C/C++.

  This topic is not expanded upon further in this document.
- Build Stages are written/configured using C/C++.

  A primary differentiator to Build Stages versus Parallel Routines invoked though a Transformer, is application; Build Stages can additionally be used as the input or output device to a DataStage Job.

Note: Because a (C/C++) Parallel Routine sits in the middle of a DataStage Job input data stream, the routine can inherit a lot of cool stuff; basically the input and output column definitions can be inferred through DataStage Run Time Column Propagation.

If you choose to create a Build Stage, *and you are truly creating a dynamic and very reusable operator*, the C/C++ code is a bit longer.

If you can create and use a Parallel Routine, many challenges have been solved; supported compilers, how to compile and link, and so on. For this reason, this document only really covers end to end creation of Parallel Routines, not creation of Build Stages.

- Wrapped Stages equate basically to operating system programs and utilities.
  - As mentioned, the March/2008 edition of this document detailed how to invoke operating system programs and utilities (OS utilities), as part of your IIS DataStage component Jobs. These OS utilities can be used

mid-stream to a DataStage Job, as well as DataStage Job input and output devices.

Basically, we used the existing DataStage Stages named; Generic, External Source, and External Target.

- The primary difference between Wrapped Stages and the External Source and External Target Stages is native parallelism.

  Wrapped Stages are natively parallel; can start numerous and concurrent input and/or output processes. External Source and External Target are serial, and only offer parallelism if you manually call to start numerous and concurrent copies.

  The Generic Stage is natively parallel.

DataStage Server Routines, using BASIC

When this document discusses the DataStage component to IBM Information Server (IIS), generally we always discuss the Enterprise Edition of DataStage (DS/EE: the fully parallel aware version of DataStage). A lower cost, and possibly less performant version of the DataStage component exists called DataStage Server. DS/EE includes the DataStage Server stages and related. DataStage Server offers an additional (base) capability in that it offer Server Routines. Server Routines are written in the BASIC programming language. While BASIC is an interpretive language, and hence executes slower and perhaps with a less optimal memory model than C/C++, BASIC is still a pretty cool and easy to learn language.

Note: Parallel Routines are written in C/C++, Server Routines are written in BASIC. Their other primary differences include; run time performance and resource consumption, and parallelism in an IBM Information Server grid configuration (grid, versus say, an SMP or single node installation). Generally Parallel Routines out perform Server Routines.

Server Routines executed within a DataStage Parallel Job can run in parallel on one SMP style node. Server Routines do not run across multiple nodes in an IBM Information Server grid configuration, as is the case with other DataStage Parallel Job Operators.

Using a DataStage Basic Transformer Stage (as opposed to the Parallel Transformer that comes with DS/EE), we can create Server Routines from BASIC language program code. And we can execute these Server Routines within DataStage Parallel Jobs.

In addition to using Server Routines through a Basic Transformer, one can also create a Shared Container which includes the Server Routine, then include the

Shared container in a Parallel Job. The May/2008 edition of IISDN, we covered how to create and work with Shared Containers.

18.2 Creating the example(s) outlined above

In this section of this document, we create 2 example IBM Information Server DataStage component Jobs;

- A DataStage Parallel Job that uses a Parallel Routine.

  As a pre-requisite to this Job, we create and compile a smallish C/C++ code fragment, that is then used by a Parallel Routine.

- A DataStage Parallel Job that uses a Server Routine.

  Server Routines are written in BASIC, so we create a smallish code fragment written in BASIC.

Both examples are really simple (Add Two Numbers), but the primary goal is achieved; How to create and configure assets of these types.

Parallel Job with Parallel Routine

To create the IBM Information Server (IIS) DataStage component Job from Figure18-1, complete the following;

1. Using your favorite source code editor or something similar (we prefer Vi), create a sample input data source.

   We created an ASCII text file with 4 columns of column type; Integer, Integer, Varchar, and VarChar. We need at least two Integer columns, because that is what we plan to operate on later; we are going to add these two column values.

2. Using your favorite C/C++ development workbench (we prefer Vi and g++), create a source code file equal to Example18-1 below.

   This file is created on the server, our example using a RedHat Linux server with a Microsoft Windows client. We saved this file with a name of, MyExternalRoutine.cpp.

Example 18-1 Sample C/C++ program code.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <apt_framework/orchestrate.h>


int MyAddTwoNumbers(int i, int j);

int main()
{

  int k;

  k = MyAddTwoNumbers(9, 8);

  printf("%d\n", k);

}

int MyAddTwoNumbers(int i, int j)
{

  return (i + j);

}
```

3. Compile this C/C++ file created above via the following steps;

   Again, this work is done on the server-

   a. We need to find the IBM Information Server program file entitled, dsenv, and we need to source it.

      If IBM Information Server is installed in,

      /opt/IBM/InformationServer

      then dsenv is located in,

      /opt/IBM/InformationServer/Server/DSEngine/dsenv

      We source this file by typing a "." at the Unix/linux prompt, then a space, and then the full pathname to dsenv. Example as shown,

      . /opt/IBM/InformationServer/Server/DSEngine/dsenv

      "Sourcing" allows us to run this program in our current shell, inheriting any environment variables which are set. We need these environment variable settings in order to compile our C/C++ program.

   b. Compile the C/C++ program we created above via the following command,

      g++ -O -c -I/opt/IBM/InformationServer/Server/PXEngine/include -Wno-deprecated MyExternalRoutine.cpp

      While the entry above line wraps in its display, that entire command should be entered on one Unix/Linux command line prompt. And that is a capital letter "O" above, not a zero or lowercase "o".

      The above still assumes that IBM Information Server is installed in,

/opt/IBM/InformationServer

The C/C++ code and the compile command listed above have a dependency on IBM Information Server library files, and as a result, environment variable settings that are specific to IBM Information Server; hence, the need to source dsenv.

4. Create an IBM Information Server (IIS) DataStage component Parallel Routine.

   a. From the Menu Bar of the DataStage Designer program, select, File -> New -> Routines -> Parallel Routine, and Click OK.

      This action produces the dialog box as displayed in Figure18-2.

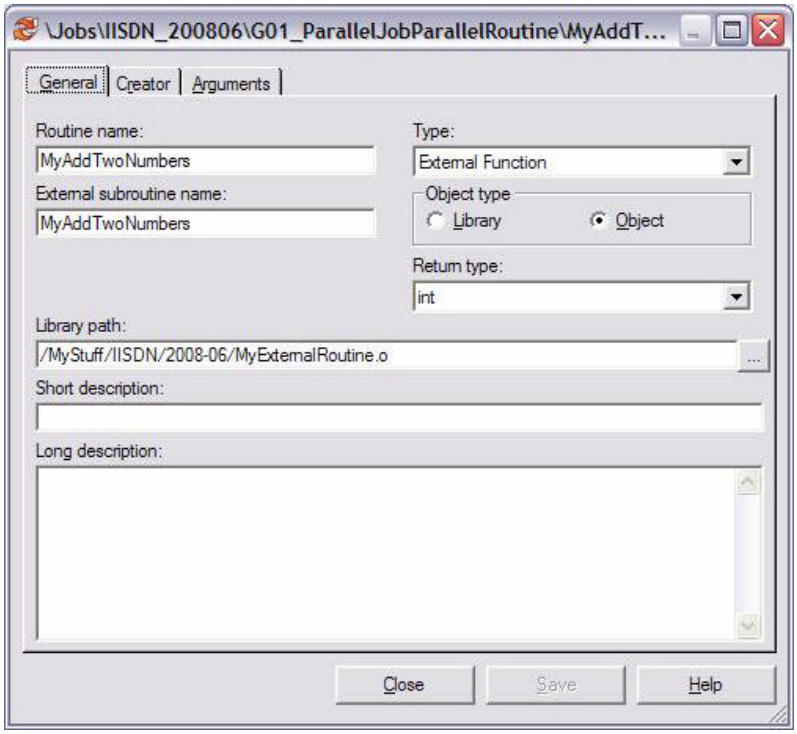

Figure 18-2 Parallel Routine, Screen 1 of 2.

   b. In the General TAB of Figure18-2, manage the display to equal what is displayed above. When you are ready, move to the Arguments TAB.

   c. In the Arguments TAB, displayed in Figure18-3, we need to add two input argument to our function; we named our arguments i and j, both of type Integer.

      When your display equals Figure18-3, Click Save, and then Click Close.

Save this new DataStage Parallel Routine in a location in the IIS Shared Repository that you can easily locate later.

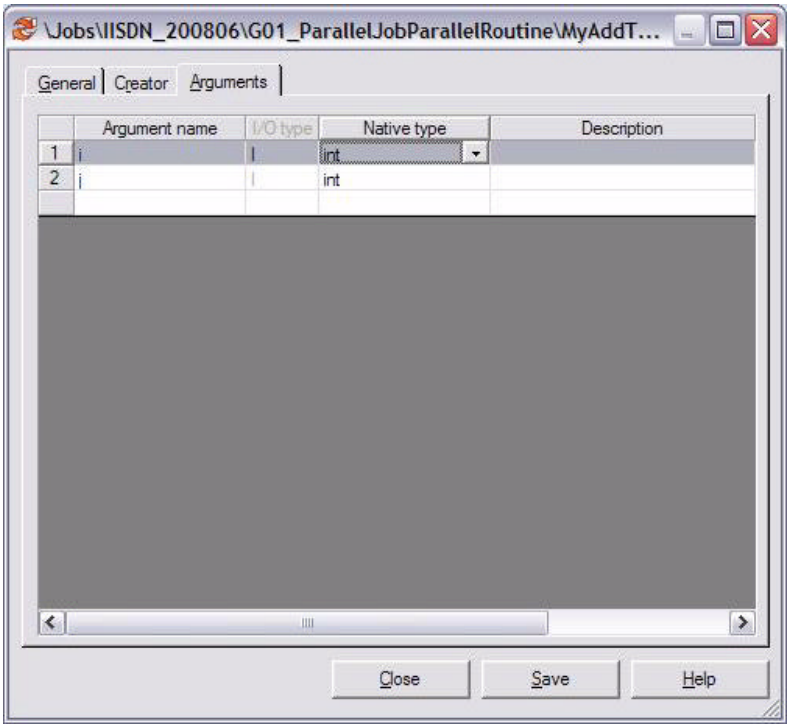

Figure 18-3 Parallel Routine, Screen 2 of 2.

5. Create the IBM Information Server (IIS) DataStage component test Parallel Job from Figure18-1.
   a. From the Menu Bar of the DataStage Designer program, select, File -> New -> Jobs -> Parallel Job.
   b. From the Palette view of the Designer program, Drag and Drop 2 Sequential File Stages and one (Parallel) Transformer Stage to the Parallel Canvas.

      Manage your display to equal what is shown in Figure18-1.

      Set all of the properties for the input and output files. The input file will read from the ASCII text file we created in Step-1 above. The output file will inherit its output columns from the Transformer Stage, you need only set the output file name and delimiter properties.
   c. Open the Transformer Stage via a Double-Click. Configure the Transformer Stage as displayed in Figure18-1.

The Derivation Expression for a new output column, we called ours col5, is listed as,

MyAddTwoNumbers(toTransformer.col1, toTransformer.col2)

And this expression is available from the Quick Pick feature of the entry field; choose first, DS Routines, and then the Input Columns to populate this control.

d. Click OK when your display equals Figure18-1.

6. Compile and Test.

   At this point, all that remains is to compile our program and test.

   A successful test reads the 4 input columns from our input data file and outputs 3 columns; columns 1 and 2 will have been added together.

Parallel Job with Server Routine

The IBM Information Server (IIS) DataStage component Job we are about to create is visually very similar in appearance to that shown in Figure18-1. To create our DataStage Parallel Job with a Server Routine, complete the following;

7. First we have to make the BASIC Transformer appear on the Palette View. By default BASIC Transformers only appear when DataStage Server Jobs are being created; as we are about to create a Parallel Job, we need to customize the Palette view.

Note: The Transformer that appears by default on DataStage Parallel Jobs is called the Parallel Transformer. The Parallel Transformer expects to run Parallel Routines, routines written in C/C++.

The Transformer that appears by default on DataStage Server Jobs is called the BASIC Transformer. The BASIC Transformer expects to run Server Routines, routines written in BASIC.

You can use Server Routines in Parallel Jobs through the BASIC Transformer; there are other techniques to accomplish this goal, this technique is easiest. When designing Parallel Jobs, the BASIC Transformer does not appear by default in the Palette view. We are about to change that.

a. Using the IIS DataStage component Designer program, open a new DataStage Parallel Job. Save this Job with a given name.

b. In the Palette view, open the Processing drawer.

   Right-Click anywhere on open space; that is, Right-Click in a location in that view that is not on top of any icon.

   In the pop up menu that is produced, select, Customization -> Customize.

This action will produce the Customize Palette dialog box.

c. In the upper left region of this dialog box, entitled, Repository Items, Click to open Stage Types -> Parallel -> Processing.

d. Select the BASIC Transformer icon with a Single-Click, then select the Right-Arrow button ("==>") to move this icon to the Current Palette groups and shortcuts items frame on the right.

e. Click OK when you are done.

8. Create a new Server Routine.

a. From the Menu Bar, select, File -> New -> Routines -> Server Routine, and Click OK.

This action produces the Server Routine (properties) dialog box, as displayed in Figure 18-4.

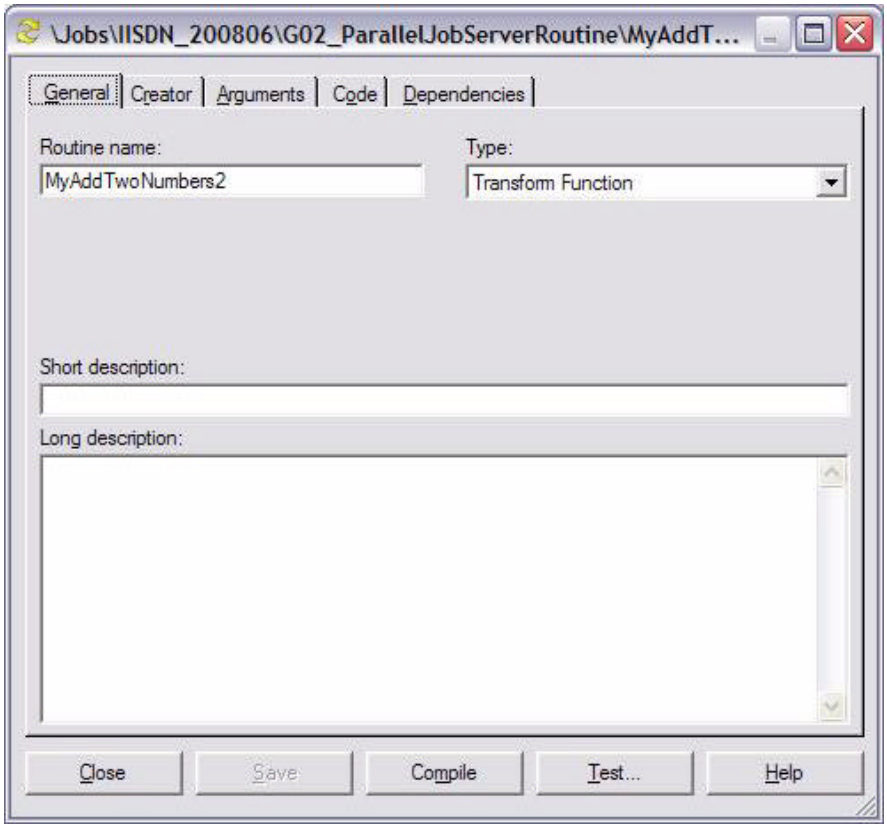

Figure 18-4 Server Routine, Screen 1 of 3.

b. In Figure18-4, and on the General TAB, give this new Server Routine a name. Also, specify that this Server Routine is of type, Transform Function.

c. Under the Arguments TAB, Figure18-5, specify two input parameters; we called our input parameters, i and j.

   As an interpretive language, BASIC does not type cast variables, so we need not specify these variables as type Integer, etcetera.

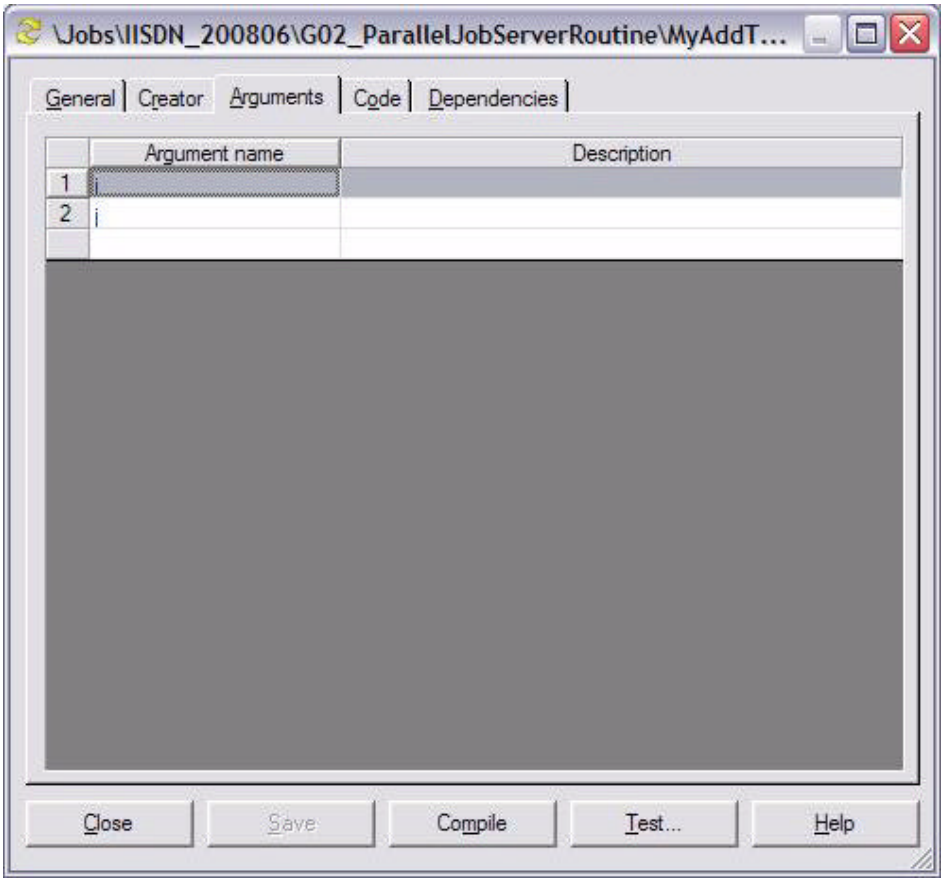

Figure 18-5 Server Routine, Screen 2 of 3.

d. In Figure18-6, and under the Code TAB, we specify the BASIC source code for our Server Routines, written in BASIC.

   We entered the BASIC assignment statement,

```
Ans = i + j ;
```


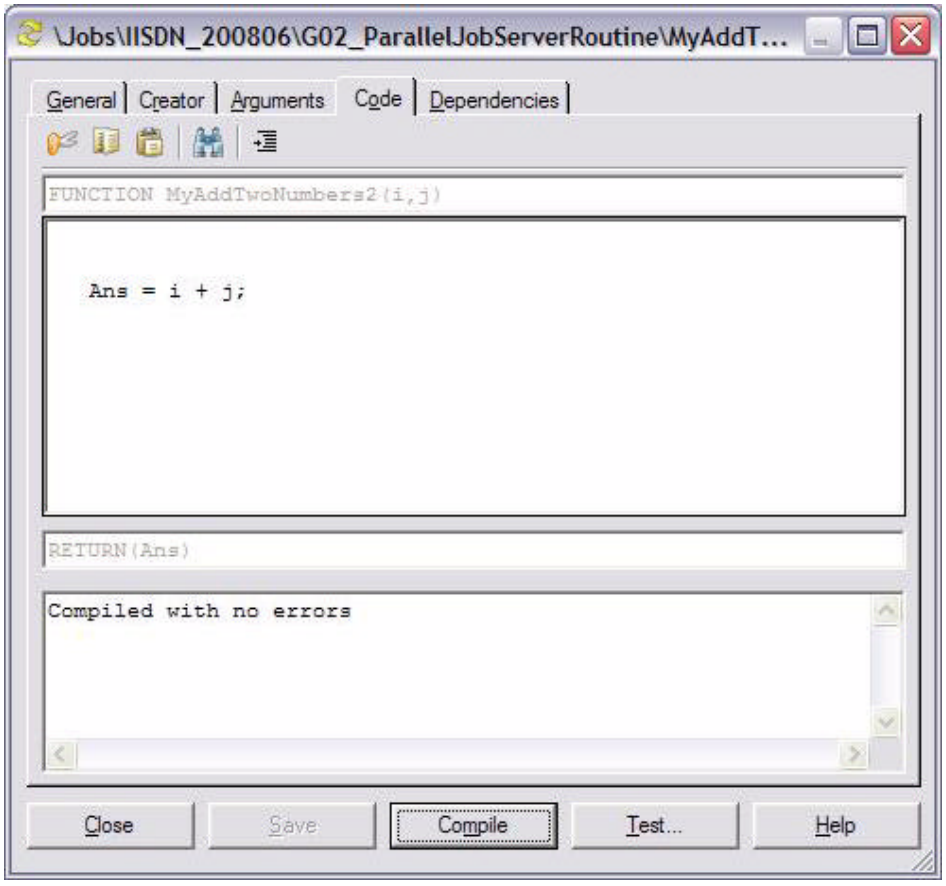

Figure 18-6 Server Routine, Screen 3 of 3.

e. When your various TABs to the Server Routines (properties) dialog box equal those in Figure18-4, Figure18-5, and Figure18-6, Click Save, then Click Compile.

   Afterwards, Click Test if you wish to test your new Server Routine.

   Click Close when you are done.

9. In the new DataStage Parallel Job you created in Step-7 above, create a Job similar to what is displayed in Figure18-1. Instead of a (Parallel) Transformer, Drag and Drop a BASIC Transformer as the central Stage.

   BASIC Transformers are only slightly different in their programming than (Parallel) Transformers, and in our current example, you wont see any noticeable differences at all.

Figure18-7 displays the configuration of the BASIC Transformer.

You are done when your Job resembles Figure 18-1 (except with a BASIC Transformer, and not a Parallel Transformer), and when your BASIC Transformer equals that as displayed in Figure18-7.

Save your Job, Compile and Test.

A successful test reads the 4 input columns from our input data file and outputs 3 columns; columns 1 and 2 will have been added together.

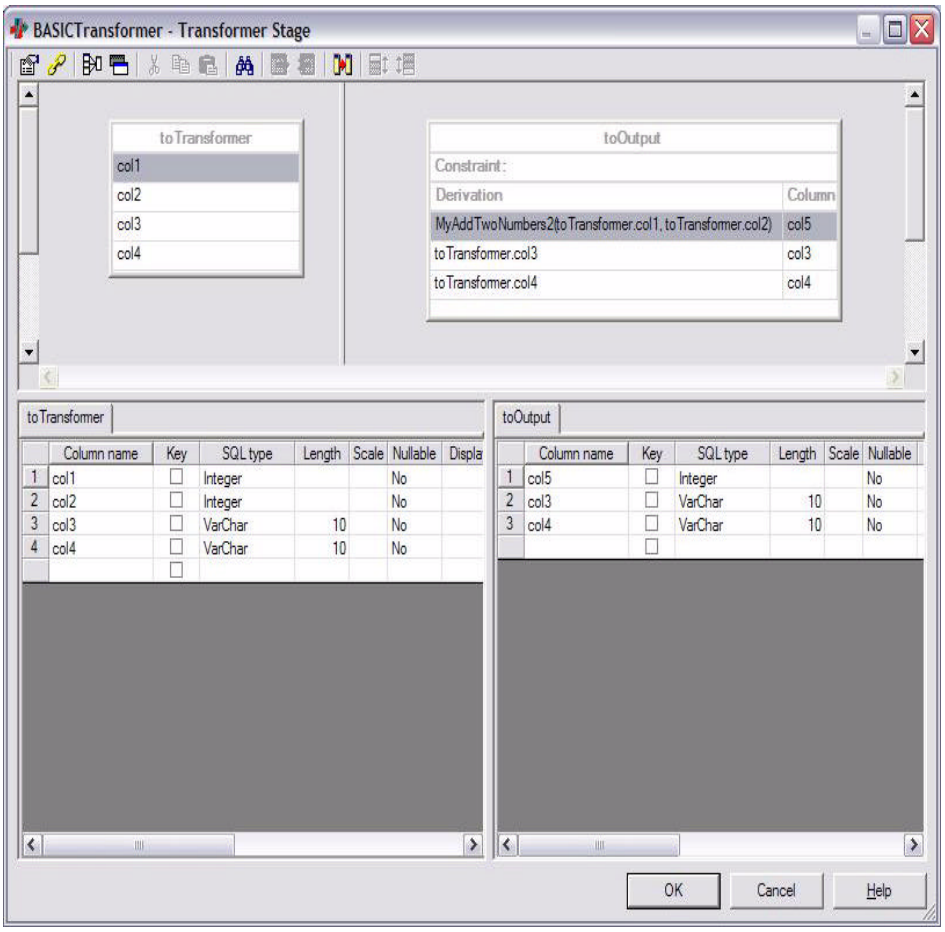

Figure 18-7 BASIC Transformer, displaying invocation of Server Routine.

18.3 Summation

In this document, we reviewed or created:

We examined some of the extensibility of the IBM Information Server (IIS) DataStage component; specifically, we created and used a Parallel Routine written in C/C++, and Server Routines written in BASIC. Also, we discussed many of the remaining options to program extensibility of IIS in other manners using C/C++, and covered briefly topics related to Java, OSH Script, and operating system commands and utilities.

Persons who help this month.

Danny 'Lieutenant Dan' Owens, Allen 'Server' Spayth, Steve Haddad, Aaron Titus, and James Wiles.

Additional resources:

None.

Legal statements:

Special attributions:

The listed trademarks of the following companies require marking and attribution:

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Microsoft trademark guidelines

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel trademark information

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

Other company, product, or service names may be trademarks or service marks of others.

