Chapter 9. September 2007

Welcome to the September 2007 edition of IBM Information Server Developer’s Notebook. This month we answer the question;

> How can I create and maintain a dual sided EDI portal with hundreds of remote sites?
>
> *While electronic data interchange (EDI) can mean different things to different folks, EDI is one of our two favorite application systems to deliver using IBM Information Server. (The other is an on line information store using Web services and SOA.)*
>
> *A classic example of an EDI portal would have, for example, a national retailing chain share store level inventory amounts per product with suppliers, so that suppliers can direct ship replacement product to each store location. In this example, the retailer normally rolls up point of sale data to a central site, where the data is cleansed, aggregated, obsfucated, and so on. The EDI system is viewed from the perspective of the retailer’s central office to each supplier, on a peer to peer basis. The retailer offers inventory data by product identifier and location, whereas the supplier offers shipment and supply chain (fulfillment) data.*
>
> *A second perspective to the above example is the point of sale data to the retailer’s central office itself. In the example we are going to follow below, the remote sites with the point of sale data are franchisees; meaning, these sites operate in agreement with the central office, but have independent and varying in store systems, hardware platforms, and related. As the original question was received, we are choosing to place a data collection agent in each remote site for reasons which are discussed below.*
>
> *To complete the examples below, you need slightly more than beginner level of capability with the DataStage EE product.*

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 version 8.01, on the Microsoft Windows XP/SP2 platform. (Variably, some of the information below is discussed for use on a Linux platform; both Windows and

Linux would offer an excellent in store platform, with the ability to automate the operation of hundreds of remote sites. While Linux is often viewed as being more complex than Windows and thus having a higher initial start up cost, Linux regularly offers a savings, especially in the area of operating dozens or more remote sites; a better economy of scale.)

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™, 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™, and Replication Server™, and Rational Data Architect™.

Obviously, IBM Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities. This month we give focus to the DataStage EE component of IBM Information Server, as well as a very brief mention of Information Analyzer, QualityStage, Rational Data Architect, Metadata Workbench, and WebSphere Information Services Director.

9.1 Topologies and run times

As the original question of operating and maintaining a dual sided EDI portal for hundreds of remote sites was received, a solution exists on many levels of description; the physical level (largely meaning the network topology), the application level (meaning what software instances are present or can be relied upon), and so on. Figure 9-1 displays one possible solution to the server placement and network topology for this solution.

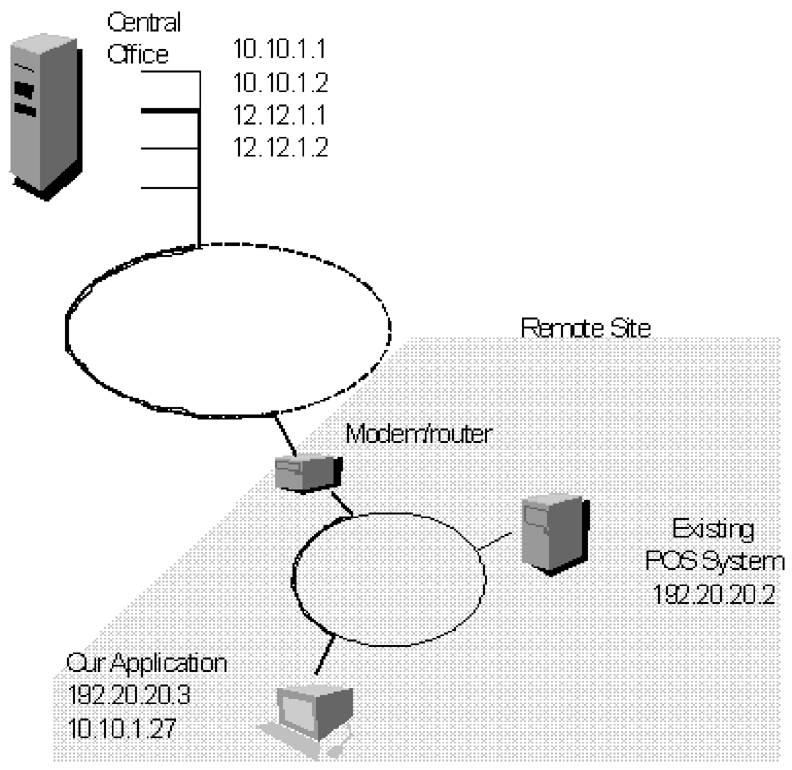


Figure 9-1   One example of server and network topology for this solution.

Concerning Figure 9-1, the following is offered:

Ê The box labeled “Existing POS System” existed before the arrival of our application. In this example, this box runs the point of sale system (a third party application) at a remote retail location.

This box could be running Windows, Linux, OS/400, or a variety of other operating systems. This box is expected to have some form of connectivity to cash registers, wireless scanners, or any other device that is supported by this third party point of sale application.

This point of sale system is expected to have a data store of some point, be it a relational database, non-relational database, flat file data storage, etcetera. It is most likely that this data store offers no documentation, or very poor documentation as to its table and column definitions (its data model). To gain an understanding of this data store’s data model, we can examine it using the *Information Analyzer component of IBM Information Server.* In effect, we build a mapping of our application’s data model, which resides at the central office, to this third party application’s data model.

The point of sale system is also expected to have some form network connectivity, preferably TCP/IP. In the case of a point of sale system that perhaps only has serial line (RS-232) communication, we could build network connectivity using Slip or PPP. If the point of sale system is entirely closed, worse case we sneaker net the data, perform manual data (re)entry, or replace or upgrade the point of sale system.

Ê The box labeled “Modem/router” offers connectivity via the Internet to the outside world, including our Central Office. First this box is a modem, because it supports the translation of our internal network TCP/IP and Ethernet communication, to whatever our Internet service provider (ISP) uses inside their network (Frame relay, whatever); expectedly, the ISP offers the remote location a DSL line of some sort.

This box is also a router, supporting a hardware firewall, DHCP, NAT, and port forwarding.

– Dynamic host configuration protocol (DHCP) allows any of the computers (network interface cards) attached to the router to be assigned a unique IP address.

  By default, these IP addresses are assigned dynamically (in a random, variable manner). Since the box supporting Our Application, and the Existing POS System are servers, we can configure the router to allow these boxes to have a fixed (static, unchanging) IP address. A static IP address is one manner that allows us to know where/how a given box may be located.

– Network address translation (NAT) means that this entire (sub)network, anything inside the remote location, can not be directly addressed (contacted) by anyone outside of the store; no one outside the store can initiate communication with computers inside the store. All of the computers inside the store share the same IP address, as viewed by the outside world.

Note: Network communication is made to *sockets*. A socket is the combination of an IP address and a port number. Sample IP addresses include, 192.20.20.2, 10.10.1.1, etcetera; basically a 4 part number, each part in the range of 1 to 255. Certain IP addresses or ranges of IP address are invalid or have special meaning. A port number is a single integer value in the range of 1 to 32,000.

The two boxes labeled “Existing POS System”, and “Our Application”, display IP addresses beginning with 192. IP addresses in this range are not addressable outside their own sub-network. In this case, its means these boxes are not accessible from the outside world.

– From the outside world, the Central Office for example, we may need to initiate contact with the box hosting Our Application. However, only the modem/router has an IP address that we can contact from the outside world.

  Port forwarding is a router supported feature that states in effect, if any outside user contacts the router IP address, and a given pre-defined port number on this router, forward that request to a box on the sub-network as though that box had received this communication directly.

  Using port forwarding on the router, we can contact the box labeled Our Application directly.

Ê The box labeled “Our Application” runs DataStage EE. By example, this server may contain two DataStage EE Projects, as detailed below;

– We call the first DataStage EE Project on this server the Administration-Project. This Project has responsibility for the following;

  • If we imagine that we only need to upload data to the Central Office daily or weekly, it could happen that this server, or the router, or some other larger scoped entity, could fail. Rather than wait for a scheduled communication with the Central Office to become late, we are going to program a heartbeat monitor; that is, we are going to ping the Central Office every 15 minutes or similar interval, so that we may detect failures more readily, hopefully before observing a loss of end user service level.

  • While we have a strategy for dynamic IP addresses within the store, within the remote location, we don’t have a strategy for a dynamic IP address being assigned to our router; maybe we can get a static IP for the router, maybe the ISP does not offer this service, or perhaps it is too costly.

As part of the data package sent with the heartbeat monitor, we will also send an amount of diagnostic data, to include; the current IP address of the router (we can gather that in real time), hard disk fullness and readiness, operational log files, and other diagnostic data.

- The Administration-Project will upload normal (end user) data packages to the central office, which is our normal use case. Also, the Administration-Project will download data packages from the Central Office, including new versions of the Application-Project, which is described next.

– The second DataStage EE Project on this server is the Application-Project proper; this is the Project containing DataStage EE Jobs to capture and aggregate data from the point of sale system, so that this data may be forwarded to the Central Office.

The primary reason to locate a DataStage EE Server installation locally within each remote site is data gathering. With DataStage EE, we can use Change Data Capture, or numerous other methods and DataStage capabilities, so that we only need to upload *new* data to the Central Office. Without a local data gathering agent (DataStage EE), we would have no means to distinguish new data from old, previously existing data. Without DataStage EE, we would have to copy the entire point of sale data store and (re)examine it each time. The (re)transmission of this potentially voluminous data store is likely to be prohibitive. Also, with DataStage EE acting as our remote data gathering agent, this whole process can be efficiently automated to include recovery and integrity mechanisms.

Ê Security, including data privacy, is a major concern of this application.

A second IP address is listed adjacent to the box entitled, "Our Application"; this IP address is displayed as 10.10.1.27. (All of these IP addresses in this document are merely examples. This address was chosen because it correlates with the example address used for the Central Office site, which also begins with 10.)

It was stated earlier that the box hosting Our Application exists behind a router, and is assigned a dynamic IP address beginning with 192. Using a virtual private network (VPN), we can tunnel an encrypted and private communication channel to the Central Office, from our remote location. This IP address appears as a virtual network interface card and any traffic that needs to go to the Central Office will automatically use this encrypted channel. (Basically all outbound traffic is encrypted automatically, without any further need to use encryption capable commands. This automatically includes any communication that DataStage EE initiates to the Central Office.)

This operating system hosting Our Application is configured to enable (connect) the VPN upon system boot.

Note: The Linux operating system includes a very capable virtual private network management package, vpnd. There is also OpenVPN (http://www.openvpn.net), a very complete and open VPN software package.

The VPN will easily handle outbound communications, from the remote location to the Central Office. For communication from the Central Office to the remote site, we can use secure shell (ssh(C)), secure ftp (sftp(C)), and other, similar commands.

Ê Scheduling, and other concerns-

The planned heartbeat monitor, normal data gathering Jobs, and other operations to be performed at the remote site can be scheduled within DataStage EE, or may use operating system scheduling capabilities. On the Linux operating system, we could use Linux Cron(A) and/or at(C) jobs.

The DataStage EE built in job scheduling has the advantage that the Application-Project could be entirely self contained; Jobs and scheduling of Jobs all in one package. Linux Cron jobs have the advantage that they operate outside of DataStage EE, and would function to include reporting errors even if the DataStage EE server were not operational.

9.2 DataStage EE in this example

Thus far we have discussed an example network and application topology (largely from the perspective of the remote site), needed to support a dual-sided EDI portal. In this section, we are going to document some of the administrative commands and capabilities needed to operate the remote site.

Creating a DataStage EE Project

Following the example used throughout this document, we expect to operate an Administrative-Project and an Application-Project; the Administrative-Project keeps the remote site running, while the Application-Project runs the jobs proper, gathering data from the point of sale application.

After initial deployment, it could happen that we need to upgrade the version of the Application-Project. We will program the Administrative-Project to create a new DataStage EE Project (a new Application-Project), operate that second Project in parallel with any earlier version of the Application-Project, and eventually delete the earlier version of the Application-Project. This and all activities will do done using an automated batch interface. Example 9-1 displays the command that will make a new, never before existing DataStage EE Project.

Example 9-1 Command to create new DataStage EE Project.

```
#
# Since it is not likely to be in our existing execution search PATH, change to the directory
# containing the “dsadmin” command. The default value for this directory is displayed below.
#

cd c:\IBM\InformationServer\Server\DSEngine\bin

#
# Execute the dsadmin command needed to create a new DataStage EE Project.
#
dsadmin -domain AAA:9080 -user UUU -password PPP -server AAA -createproject CCC
```

Concerning Example 9-1, the following is offered:

Ê The lines beginning with a pound symbol (#), are comment lines and do not execute.

Ê We change to the directory where the “dsadmin” command is located. Alternately we could have added this directory to our PATH environment variable, or prefaced the dsadmin command with this value.

Ê Arguments to the dsadmin command as we need it to execute include:

- AAA is the name of the operating system host, and this value is used twice in our command string. On Linux, this is the Linux host name.
- 9080 is the port number that the DataStage EE server is listening on. 9080 is the default value.
- UUU is the username capable of creating a new DataStage EE Project.
- PPP is the password to authenticate user UUU.
- CCC is the name of the new DataStage EE Project to create.

Ê This command can take several minutes to execute. After completion, the newly created Project is empty; it has no Jobs, Sequences, or related.

Populating a DataStage EE Project

Per the example we are following, we are going to deploy our Application-Project as a whole unit; meaning, we deploy entirely new DataStage EE Projects not subsets of Projects. Alternately, we could deploy the deltas alone; meaning, we could deploy only the DataStage EE Jobs that have changed release over release. Each approach has its advantages. By deploying whole Projects, we are more certain that we wont be leaving any forgotten Jobs or resources.

In DataStage EE parlance, we populate a DataStage EE Project by importing a previously exported DataStage EE DSX file; a DSX file may contains any assortment of Jobs, Table Definitions, and so on. To create a DataStage EE DSX (DataStage Exchange) file, complete the following;

1. Use the DataStage EE Designer to connect to any available DataStage EE Project. Connect to a Project containing at least one job, or create a simple Job at this time.
2. From the menu bar, select: Export -> DataStage Components.
3. In the dialog box that is produced, Click the Add button to browse and select any number of DataStage resources. Be certain to select at least one compiled and runnable DataStage EE Job.
4. Be certain to Check the visual control entitled, "Include dependent items".
5. Specify an output destination by populating the visual control entitled, "Export to file". The standard file suffix for this type of resource is, "dsx".
6. Click Export to complete this activity.

Example 9-2 displays the command that calls to import DataStage EE components to an existing DataStage EE Project.

Example 9-2   Command to import DataStage EE components to a Project.

```
#
# Since it is not likely to be in our existing execution search PATH, change to the directory
# containing the "dscmdimport" command. The default value for this directory is displayed below.
#

cd c:\IBM\InformationServer\Clients\Classic

#
# Execute the dscmdimport command needed to import DataStage EE Project resources.
# (Jobs, Sequences, etcetera.)
#
dscmdimport /D=AAA:9080 /U=UUU /P=PPP /H=AAA /NUA NewProject JJJ
```

Concerning Example 9-2, the following is offered:

Ê The lines beginning with a pound symbol (#), are comment lines and do not execute.

Ê We change to the directory where the "dscmdimport" command is located. Alternately we could have added this directory to our PATH environment variable, or prefaced the dsadmin command with this value.

Ê Arguments to the dscmdimport command as we need it to execute include:

- AAA is the name of the operating system host, and this value is used twice in our command string. On Linux, this is the Linux host name.
- 9080 is the port number that the DataStage EE server is listening on. 9080 is the default value.
- UUU is the username capable of importing DataStage EE Project resources.
- PPP is the password to authenticate user UUU.
- JJJ is the absolute or relative pathname to the DSX file containing the DataStage EE resources to import.

Ê If the operating system type and version, and DataStage EE software version, match between the box creating the DSX file and then importing the DSX file match, we wont need to explicitly compile these newly imported Jobs; these Jobs are ready to run.

Running a DataStage EE Job

At this point, our DataStage EE Project is fully ready to run its Jobs. Example 9-3 displays the command that calls to run a given DataStage EE Job or Sequence.

Example 9-3 Command to run a DataStage EE Job or Sequence.

```
#
# Since it is not likely to be in our existing execution search PATH, change to the directory
# containing the “dsjob” command. The default value for this directory is displayed below.
#

cd c:\IBM\InformationServer\Server\DSEngine\bin

#
# Execute the dsjob command needed to run a given DataStage EE Job or Sequence.
#
dsjob -domain AAA:9080 -user UUU -password PPP -server AAA -run -jobstatus EEE FFF
```

Concerning Example 9-3, the following is offered:

Ê The lines beginning with a pound symbol (#), are comment lines and do not execute.

Ê We change to the directory where the “dsjob” command is located. Alternately we could have added this directory to our PATH environment variable, or prefaced the dsadmin command with this value.

Ê Arguments to the dsjob command as we need it to execute include:

- AAA is the name of the operating system host, and this value is used twice in our command string. On Linux, this is the Linux host name.
- 9080 is the port number that the DataStage EE server is listening on. 9080 is the default value.
- UUU is the username capable of importing DataStage EE Project resources.
- PPP is the password to authenticate user UUU.
- EEE is the name of the DataStage EE Project which contains the Job we want to run.
- FFF is the name of the DataStage EE Job itself.

Retrieving the DataStage EE log for a given Job

The return (exit) code from the "dsjob" command will equal 1 on receipt of a warning, 2 for error. If we should need to review the Activity Log for a given DataStage EE Job, we execute another form of the dsjob command, as detailed below in Example 9-4.

Example 9-4 Retrieving a DataStage EE Job Activity Log.

```
#
# Since it is not likely to be in our existing execution search PATH, change to the directory
# containing the "dsjob" command. The default value for this directory is displayed below.
#

cd c:\IBM\InformationServer\Server\DSEngine\bin

#
# Execute the dsadmin command needed to retrieve a given DataStage EE Job or Sequence
# Activity Log file.
#
dsjob -domain AAA:9080 -user UUU -password PPP -server AAA -logdetail EEE FFF
```

All of the variable arguments to the dsjob command in Example 9-4, equal those from Example 9-3; meaning, EEE is the Project Name, FFF is the Job name, and so on.

Deleting a DataStage EE Project, getting a list of Projects

After the new version of our DataStage EE Application-Project has passed trial, running concurrently with any earlier version of the Application-Project, we can delete the older version of the Project. Example 9-5 lists the command to delete a DataStage Project, and also to produce a dynamic list of all Projects contained in a given DataStage EE server.

Example 9-5   Deleting a DataStage EE Project, getting a list of all Projects.

```
#
# Since it is not likely to be in our existing execution search PATH, change to the directory
# containing the "dsadmin" command. The default value for this directory is displayed below.
#

cd c:\IBM\InformationServer\Server\DSEngine\bin

#
# Execute the dsadmin command needed to create a new DataStage EE Project.
#
dsadmin -domain AAA:9080 -user UUU -password PPP -server AAA -deleteproject CCC

#
# Get a dynamic list of Projects contained in a given DataStage EE Server.
#

dsadmin -domain AAA:9080 -user UUU -password PPP -server AAA -listprojects
```

All of the variable arguments to the dsjob command in Example 9-5, equal those from Example 9-1; meaning, CCC is the Project Name, AAA is the host name, and so on.

9.3 Other Information Server components

As this example in this document is constructed, we are delivering a dual sided electronic data interchange (EDI) portal with hundreds of sites. While most of our attention has been given to the remote site where we expect to run DataStage EE, the following is also offered:

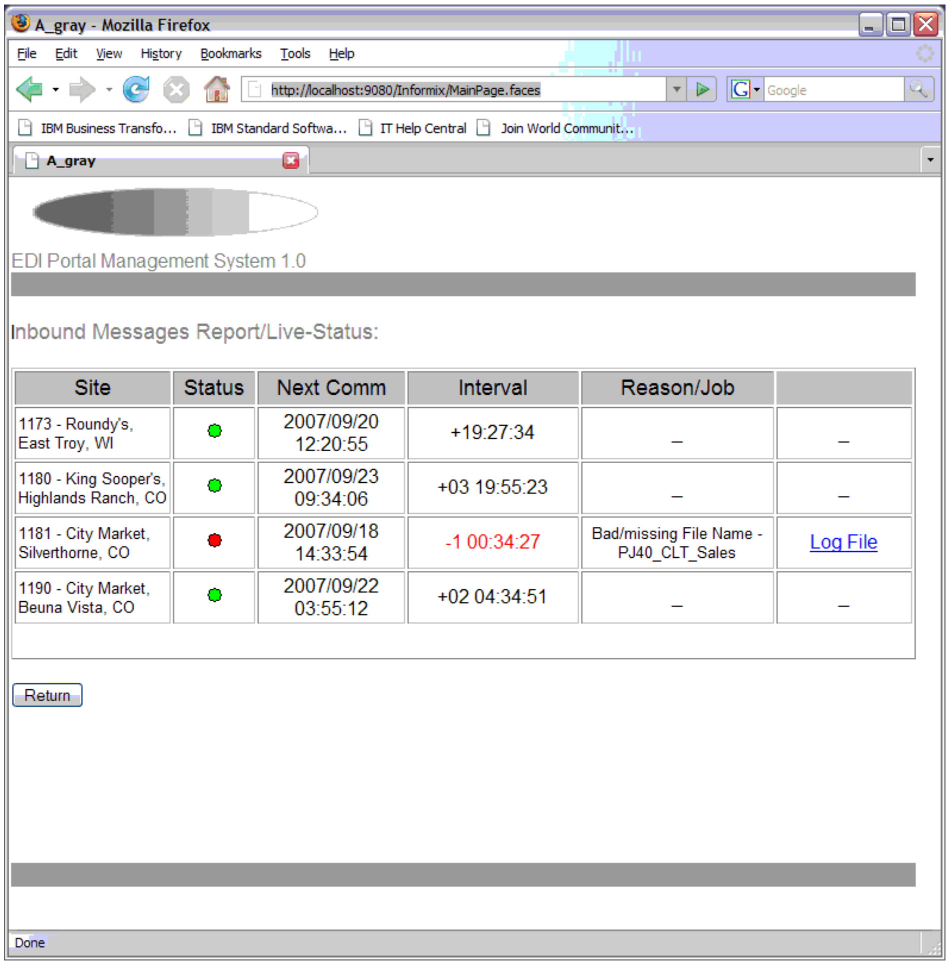
Ê If the requested application service levels allow for it, we can use scheduled file transfer protocol (ftp(C)) DataStage EE operators and Jobs to move data from the remote sites to the Central Office. This solution requires a DataStage EE at each of the remote sites. We ftp from the remote site to the Central Office, and secure shell and secure ftp from the Central Office to any remote site, should any runtime requirements change without the ability to anticipate them.

  If we need more of a real time capability to send and retrieve data to and from remote sites and the Central Office, we can easily use the WebSphere Information Services Director (WISD) component to IBM Information Server.

this path places a J2EE compliant application server at each of the remote sites. Data transfer to and from the remote sites could continue with ftp, or could be moved to HTTP, both via secure (encrypted) communication channels.

- We anticipate using DataStage EE at each remote site to aggregate data using Change Data Capture, or other DataStage EE capabilities.

  The QualityStage component to IBM Information Server would be used at the Central Office to cleanse and standardize data; for example, if we are retrieving product that is not readily identified by a standard code (instead it is identified by a slightly variable textual description), QualityStage can fix that problem for us.
- As mentioned previously, this example moves data from a third party application where that application data model is not known; the Information Analyzer component to IBM Information Server solves that problem, allowing us to capture meta data on the received data set to include; column definitions, column formats and ranges, and related.
- The Rational Data Architect component to IBM Information Server would allow us to take meta data generated from Information Analyzer, and realize this metadata as SQL constructs; SQL DDL statements, and more.
- And finally, the MetaData Workbench component would allow us to support a full traceability of any data element found to exist in side this entire application landscape; from any given data column, how is this data sourced, enhanced, corrected, basically detailing the entire life-cycle of any individual data element.
- The XMeta database, the SQL database that supports the shared metadata between each of the IBM Information Server components, is used in part to aggregate data, and to support a simple HTML based to monitor and manage each of the dozens or hundreds of remote sites. Example of such an application is shown in Figure 9-2.


A_gray - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:9080/Informix/MainPage.faces
IBM Business Transfo... IBM Standard Softwa... IT Help Central Join World Communit...
A_gray
Return
Done

EDI Portal Management System 1.0

Inbound Messages Report/Live-Status:

Site	Status	Next Comm	Interval	Reason/Job	
1173 - Roundy's, East Troy, WI	🟢	2007/09/20 12:20:55	+19:27:34	–	–
1180 - King Sooper's, Highlands Ranch, CO	🟢	2007/09/23 09:34:06	+03 19:55:23	–	–
1181 - City Market, Silverthorne, CO	🔴	2007/09/18 14:33:54	-1 00:34:27	Bad/missing File Name - PJ40_CLT_Sales	Log File
1190 - City Market, Beuna Vista, CO	🟢	2007/09/22 03:55:12	+02 04:34:51	–	–

Figure 9-2 Sample custom application to aggregate data from dozens or hundreds of remote sites.

9.4 Summation

In this document, we reviewed or created:

Essentially we had an architectural discussion of one of our favorite IBM Information Server applications, delivering and supporting an EDI portal. In addition to the architectural discussion, we covered the technology to automate the batch operations for creating and deleting DataStage EE Projects, running DataStage EE Jobs and Sequences, and gathering diagnostic data of same.

IBM offers many hardware and software products for EDI, including IBM Partner Gateway, and EDI and B2B solution. See,

> http://www-304.ibm.com/jct03002c/software/integration/wspartnergateway/enterprise/index.html

IBM Partner Gateway is solution software. What we have discussed in this document uses IBM Information Server as the supporting technology to deliver an EDI portal of one’s one design.

Additional resources:

None.

Thank you to the persons who helped this month:

Wesley Williams.

Legal statements:

Special attributions: