

Data Exchange between Oracle EBS & Other Systems – the Easy Way

A guide to automating data exchange and event-driven processes from within Oracle EBS

Electronic exchange of data encompasses most areas of business, including, but not limited to, manufacturing, medical, banking/financial, government, and retail organizations. Regardless of the type of business, automated exchange of mission-critical data between and within organizations and their computer systems has become a requirement for modern day commerce.

While Oracle EBS addresses most technical needs of today's companies the exchange of data between Oracle EBS and third-party systems is an area that requires much consideration. For example, third-party systems produce files that are needed by Oracle EBS, and Oracle EBS produces files that are needed by third-party systems. Unfortunately, Oracle EBS cannot possibly have a "hook" into every third-party system for the exchange of data. As a result, companies often invest significant time, money, and resources to write custom scripts or programs automating this needed data exchange. Such activities draw valuable time away from employee productivity, are an unnecessary expense and create ongoing costs with maintenance.

Data Interchange Requirements

Given the above, companies are looking for solutions that seamlessly handle the interchange of data between Oracle EBS and a third-party system, allowing an organization to concentrate on normal business functions. Consider some of the requirements of such a system:

- Automatic exchange of data between Oracle EBS and ANY third-party system
- No customizations, programming, or scripts required
- Staff knowledge of other third-party systems is NOT necessary
- Interface within Oracle EBS is easy to setup and maintain
- Solution is native to Oracle EBS, avoiding the need to learn a new interface
- UNIX Administrators (and other specialized IT resources) are not required to operate the solution
- Rules for processing of data exchange can be easily changed/adapted to meet business needs
- Automated email notification based upon defined rules
- Creation of concurrent requests in Oracle EBS, triggered by an event, not a pre-defined schedule

Actions, Events, Rules, and Notifications

To further explore such requirements, consider the following scenario using a company called We-Sell-Stuff, Inc. We-Sell-Stuff is a growing retail organization with two retail stores and a main headquarters location. The retail stores use POS terminals for transactions related to the selling of their goods. These POS terminals are connected to a server at each retail location that stores files related to the transactions. These files accumulate on their respective servers throughout the day. By the end of the day, these files need to be imported into the Oracle EBS server located at the main headquarters. The importing of this data takes place within Oracle EBS by generating a concurrent request.

Additionally, the POS terminals must be updated once per week with product information, such as pricing, inventory, etc. This is accomplished by running a concurrent request in Oracle EBS to generate a file containing the updated information. This information must then be deposited onto the servers at each retail location, where an application updates the POS terminals.

For the processing of transaction files, We-Sell-Stuff presently uses a process whereby an employee manually checks for the depositing of the transaction files onto each server after 10 PM every day. Upon confirmation that all of the files are located on the server, the employee manually FTPs the files from their respective servers to the server at the headquarters location. The employee then runs a concurrent request called "Process Today's Transactions" in Oracle EBS to process the transaction files.

The current process in place for the updating of transaction files is draining resources from the staff, both at headquarters as well as the individual stores. Someone from the headquarters location has to be available at 10 PM every night to transfer the transaction files to the system. This is inconvenient enough, but the scenario worsens and more resources have to be engaged if all of the files from the 10 AM, 2 PM, 6 PM and 10 PM batches are NOT on each store's server. In that case, phone calls are made late at night to find out what happened to the files. Additionally, the application that writes the transaction batches can sometimes output duplicate files for any given batch, creating a real problem. Someone must ensure that only one batch was written to the directory until such time as the defect in the application can be tracked down and corrected. Otherwise, duplicate transactions are imported into Oracle EBS, resulting in significant confusion and wasted effort.

For the processing of updates to the POS terminal data, We-Sell-Stuff presently runs a concurrent request in Oracle EBS to generate the files. They then use a UNIX-based "cron" (time-based job scheduler) process to FTP the file to the server at each retail location where an application runs to update the POS terminal.

The current process in place for the weekly POS updates, although automated, still pulls significant time and resources from the IT staff. The cron job requires constant maintenance, especially since the FTP passwords are changed frequently. Additionally, there is no "audit trail" of what happened when the FTP attempt was made; instead, a manual check for the presence of the files on each server is required to ensure that the process was successful.

Business is "booming" for We-Sell-Stuff and as a result, a new store (store #3) is in the process of being constructed, scheduled to open within the next two months. This fast-growing company needs to have their staff focused on the opening of the new store, and therefore cannot continue to devote these resources to the transferring of files between their applications and Oracle EBS. They need a solution that will easily implement into their present environment of two stores, as well as accommodate future growth.

Now that we have set the stage for our customer's problem, let's evaluate each of these data exchange processes further, defining them in terms of events, notifications, rules, and actions. An event is something that takes place (e.g., a file is found in a specific directory on a specific host) to trigger an action (e.g., the file is then FTP'd to another server). Rules are setup to control actions (e.g., do not perform this action due to a duplicate file). Notifications are setup to deliver messages via email regarding a specific event (e.g., notify if the file is not present in the directory).

Daily Transactions

Events: Transaction files are deposited onto the retail store's server (WSS-1 for store 1 and WSS-2 for store 2) at intervals (10 AM, 2 PM, 6 PM, and 10 PM) in a directory named /home/transactions/daily_processing. The files are named using a convention whereby the name of the file denotes the store number, date (YYYYMMDD), and the (military) time of the file's deposit. For example, Store 1 would have files that looked like this: store1_20100420_transactions_10*.txt for the 10 AM batch, store1_20100420_transactions_14*.txt for the 2 PM batch, store1_20100420_transactions_18*.txt for the 6 PM batch, and store1_20100420_transactions_22*.txt for the 10 PM batch.

Notifications: Headquarters needs to be notified via email if not all files are present in the /home/transactions/daily_processing directory after each of the time intervals (10 AM, 2 PM, 6 PM, and 10 PM) for each store.

Rules: Due to a bug in the POS application, it is possible for duplicate batches to be written to the /home/transactions/daily_processing directory. It is critical to ensure that if this occurs, the second and any subsequent duplicate files do NOT get processed.

Actions: The /home/transactions/daily_processing directory on WSS-1 and WSS-2 need to be scanned for files matching the patterns for each batch and FTP'd to the Oracle EBS server (HQ-1) located at headquarters. Once the files have been FTP'd to headquarters, a concurrent process needs to be initiated in Oracle EBS called "Process Today's Transactions".

POS Weekly Updates

Events: An Oracle EBS Concurrent Request is run that deposits all POS update files into a local directory, /home/pos_updates on the Oracle EBS HQ-1 server. The Concurrent Request is run every Saturday at 9 PM, and usually takes approximately one hour to complete.

Notification: In the event that the update files cannot be generated, headquarters needs to be notified via email.

Actions: The local /home/pos_updates directory on HQ-1 needs to be scanned for update files matching the pattern of *.upd and FTP'd to the /home/pos_updates directory on each server (WSS-1 and WSS-2).

Automating the Data Interchange Process

We-Sell-Stuff presently has a need for automating the exchange of data between Oracle EBS and a third-party system in the following areas: Daily Transactions and Weekly POS Updates. At this point, we have broken down the processes into events, notifications, rules, and actions and need a solution that addresses all of those requirements.

The solution should easily exchange data between Oracle EBS and ANY third-party system, and should be native to Oracle EBS, therefore easy to use, maintain, and intuitive. Ideally, the software should be able to break the processes down into logical components, as previously defined, by events, notifications, rules, and actions.

The remainder of this paper uses DataZing, a product of STR Software, as a third-party solution to completely address all of the requirements needed for the automatic exchange of data between Oracle EBS and third-party systems. DataZing allows for the configuration of events, notifications, rules, and actions, all native within Oracle EBS. DataZing is capable of monitoring local or remote directories, and based upon a matching file pattern, can email, FTP, or submit an Oracle EBS Concurrent Request. Rules can be applied that are checked before performing actions, and email notifications can be configured based upon these rules or events.

Let's explore how DataZing automates the processing needs for daily transactions and POS updates at We-Sell-Stuff, Inc.

DataZing Processing of Daily Transactions at We-Sell-Stuff, Inc.

Defining Events

We will be using the DataZing Event Configuration form in Oracle EBS to define our previously outlined events. The DataZing Event Configuration form allows for individual events to be defined, and ties notifications, processing rules, and actions to each event.

DataZing - Event Configuration Form

Event

Event Type: Remote File Search (FTP)

Event Seq: 1

Event Description: Search for 10 AM Batches from Store 1. When found, submit concurrent request.

Remote Host: WSS-1

Username: transmgr

Password: *****

Remote Directory: /home/transactions/daily_processing

File Pattern: store1_*_transactions_10*.txt

[View Events in Sequence](#)

Notifications | Processing Rules | Actions

Seq	Notification Type	Description
1	Notify if event does not occur by...	Notify administrator if 10 AM batch is not present

[Create/View Notification](#)

[Synchronize Config](#)

Recall that there are four different times throughout the day that batches are generated by the POS terminal applications for the depositing of transaction files onto each store's local server. Using the DataZing Event Configuration form, we need to define four events per store, one per time slot per store, to properly handle the exchange of data between the servers at each store and the Oracle EBS server at headquarters.

The event type for all events defined for transaction processing is "Remote File Search (FTP)". This indicates that a remote directory will be scanned at a configurable interval of time. Other possible event types are local file search and secure remote file search (SFTP). Different information must be provided based upon the type of event. For example, "Remote File Search (FTP)" requires login information, the directory to scan, and the file pattern.

The screenshots below show the four events for store 1 for the 10 AM batch, 2 PM batch, 6 PM batch, and 10 PM batch. The top portion of the form defines the event. The notifications, processing rules, and actions associated with each event are configurable in the lower part of the form, as discussed next. Of course, the setup for store 2 would be identical except for the remote host and possibly the login information.

DataZing - Event Configuration Form

Event

Event Type: Remote File Search (FTP)

Event Seq: 1

Event Description: Search for 10 AM Batches from Store 1. When found, submit concurrent request.

Remote Host: WSS-1

Username: transmgr

Password: *****

Remote Directory: /home/transactions/daily_processing

File Pattern: store1_*_transactions_10*.txt

[View Events in Sequence](#)

Notifications | Processing Rules | Actions

Seq	Notification Type	Description
1	Notify if event does not occur by...	Notify administrator if 10 AM batch is not present

[Create/View Notification](#)

[Synchronize Config](#)

DataZing - Event Configuration Form

Event

Event Type: Remote File Search (FTP)

Event Seq: 2

Event Description: Search for 2 PM Batches from Store 1. When found, submit concurrent request.

Remote Host: WSS-1

Username: transmgr

Password: *****

Remote Directory: /home/transactions/daily_processing

File Pattern: store1_*_transactions_14*.txt

[View Events in Sequence](#)

Notifications | Processing Rules | Actions

Seq	Notification Type	Description
1	Notify if event does not occur by...	Notify administrator if 2 PM batch is not present

[Create/View Notification](#)

[Synchronize Config](#)

Defining Notifications

Remember the requirement to notify the administrator at headquarters if something “goes wrong” and the batch for this time slot is not generated. The Notifications tab of the DataZing Event Configuration form is used to define such notifications. There are several types of notifications available, including notification if an event occurs within a certain time period, notification if an event does NOT occur by a certain time, and notification if a duplicate event occurs within a time period.

For each of the four events at store 1 (as well as the four events for store 2), a notification should be configured to alert the administrator if the batch for that time period is not present within fifteen minutes of its expected deposit.

The example above shows the definition of a notification for the 10 AM batch. If the event does not occur by 10:15 AM, a notification email is to be sent to the administrator at headquarters alerting the IT staff of a problem. Of course, the notification definition will have a different time for each batch (e.g., the 2 PM batch will have a time of 2:15 PM, etc.)

Defining Processing Rules

Due to a bug in the POS application, it is possible for duplicate batches to be written to the /home/transactions/daily_processing directory. It is critical to ensure that if this occurs, the second and any subsequent duplicate files do NOT get processed. The Processing Rules tab of the DataZing Event Configuration form allows for processing rules such as this to be defined. In this example, if a duplicate file appears in the /home/transactions/daily_processing directory on WSS-1 (store 1) for the 10 AM batch, the processing will NOT be performed, and the duplicate file is deleted by DataZing. This processing rule will need to be defined for each of the four events for each store.

The screenshot shows the DataZing - Event Configuration Form. The main window has tabs for Notifications, Processing Rules, and Actions. The Event tab is active, showing fields for Event Type (Remote File Search (FTP)), Event Seq (1), Event Description (Search for 10 AM Batches from Store 1. When found, submit concurrent request.), Remote Host (WSS-1), Username (transmgr), Password (*****), Remote Directory (/home/transactions/daily_processing), and File Pattern (store1_*_transactions_10*.txt). A button labeled 'View Events in Sequence' is also present.

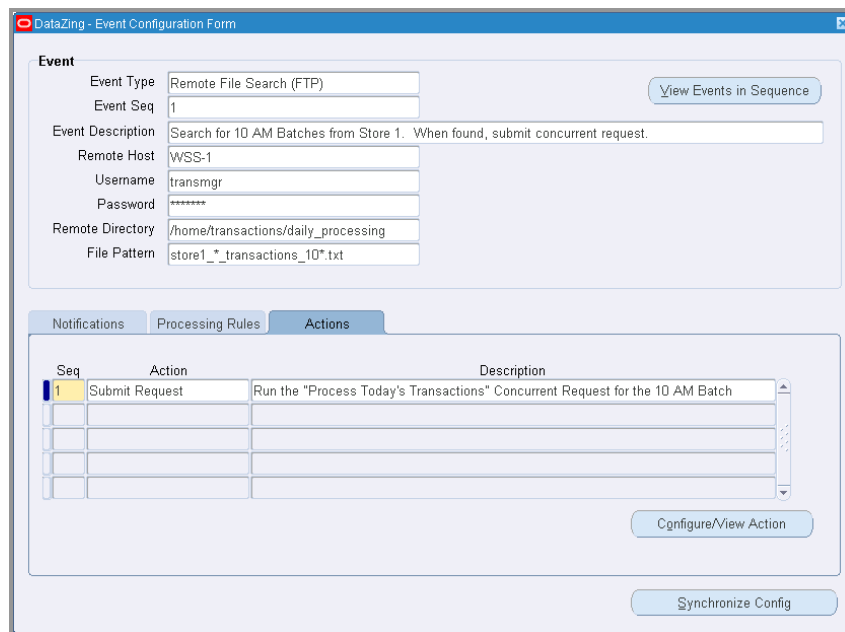
The Processing Rules tab is also visible, showing a table with columns for Seq, Rule Type, and Don't process. The first row shows Seq 1, Rule Type 'Do not process if duplicate event occurs between...', and Don't process.

A smaller window titled 'DataZing - Event Configuration Form' is overlaid on the main window, showing the Rule Definition section. It includes a checkbox for 'Do not process if duplicate event occurs between...', a time selection area (8:00 AM to 10:00 PM), a calendar for 'on date(s) of every month', and a section for 'and/or days of every week' with a day selection grid (S, M, T, W, T, F, S). There is also a checkbox for 'Notify if rule prevents processing?' and a Notification Options section with fields for From Name, From Email, Subject, To, and Message.

Defining Actions

Lastly, actions can be defined based upon the occurrence of an event. DataZing can perform different types of actions, including Email, FTP, Submit Request to Oracle EBS, Submit Request Set to Oracle EBS, and even moving or renaming a file.

For our transaction processing, an action is defined via the Actions tab of the DataZing Event Configuration form to submit the "Process Today's Transactions" concurrent request. This action will need to be defined for each batch for each store.



Event

Event Type: Remote File Search (FTP)

Event Seq: 1

Event Description: Search for 10 AM Batches from Store 1. When found, submit concurrent request.

Remote Host: WSS-1

Username: transmgr

Password: *****

Remote Directory: /home/transactions/daily_processing

File Pattern: store1_*_transactions_10*.txt

[View Events in Sequence](#)

Actions

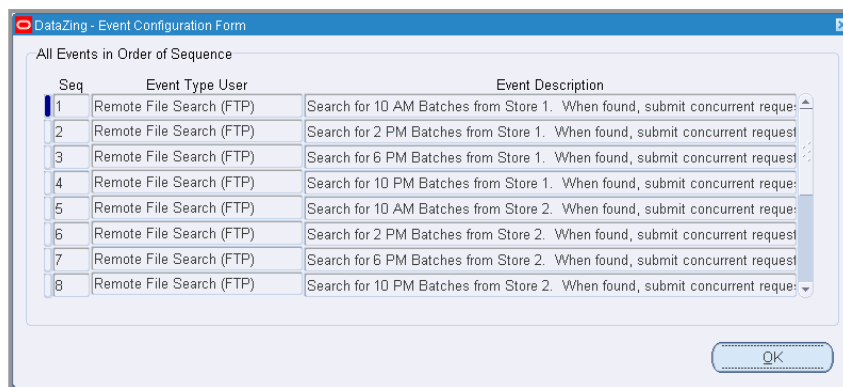
Seq	Action	Description
1	Submit Request	Run the "Process Today's Transactions" Concurrent Request for the 10 AM Batch

[Configure/View Action](#)

[Synchronize Config](#)

Viewing Configured Events

You can also view all configured events using the DataZing Event Configuration form. The example below shows all events related to transactions (in order of their sequence) for We-Sell-Stuff, Inc.



All Events in Order of Sequence

Seq	Event Type	User	Event Description
1	Remote File Search (FTP)		Search for 10 AM Batches from Store 1. When found, submit concurrent request
2	Remote File Search (FTP)		Search for 2 PM Batches from Store 1. When found, submit concurrent request
3	Remote File Search (FTP)		Search for 6 PM Batches from Store 1. When found, submit concurrent request
4	Remote File Search (FTP)		Search for 10 PM Batches from Store 1. When found, submit concurrent request
5	Remote File Search (FTP)		Search for 10 AM Batches from Store 2. When found, submit concurrent request
6	Remote File Search (FTP)		Search for 2 PM Batches from Store 2. When found, submit concurrent request
7	Remote File Search (FTP)		Search for 6 PM Batches from Store 2. When found, submit concurrent request
8	Remote File Search (FTP)		Search for 10 PM Batches from Store 2. When found, submit concurrent request

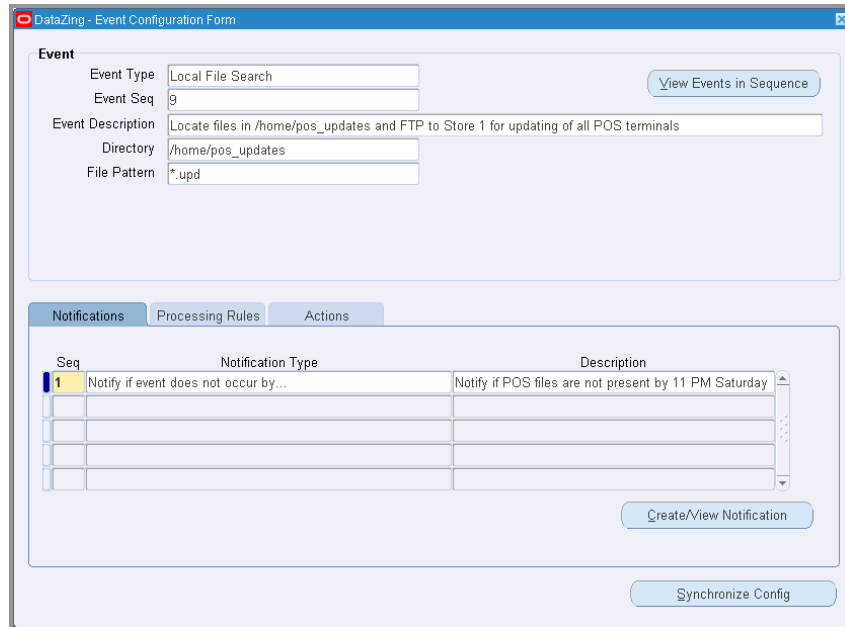
[OK](#)

DataZing Processing of POS Weekly Updates at We-Sell-Stuff, Inc.

Defining Events

In addition to daily transactions, weekly updates regarding product and pricing information are posted to each store's server where an application then runs to update the POS terminals. An Oracle EBS Concurrent Request is run that deposits all POS update files into a local directory, /home/pos_updates on the Oracle EBS HQ-1 server. The Concurrent Request is run every Saturday at 9 PM, and usually takes approximately one hour to complete.

Using the DataZing Event Configuration form, the Event to handle this scenario is defined as follows:

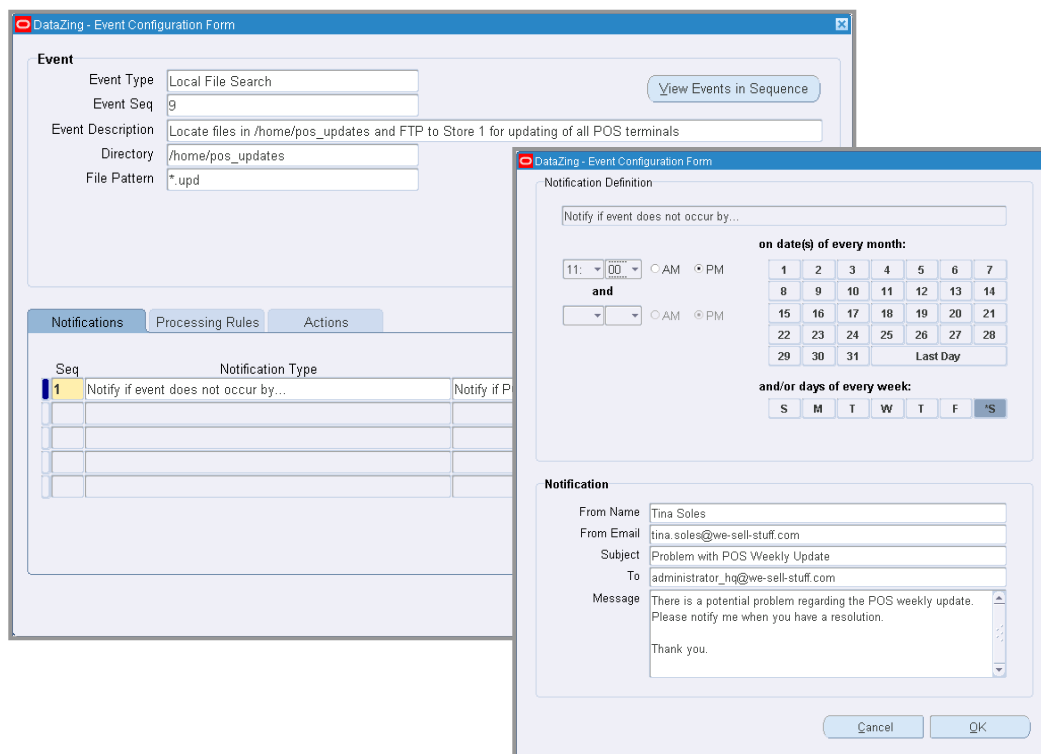


The screenshot shows the 'DataZing - Event Configuration Form'. The 'Event' tab is active. The 'Event Type' is 'Local File Search', 'Event Seq' is '9', 'Event Description' is 'Locate files in /home/pos_updates and FTP to Store 1 for updating of all POS terminals', 'Directory' is '/home/pos_updates', and 'File Pattern' is '*.upd'. A 'View Events in Sequence' button is present. Below the 'Event' tab are 'Notifications', 'Processing Rules', and 'Actions' tabs. The 'Notifications' tab shows a table with one row: Seq 1, Notification Type 'Notify if event does not occur by...', and Description 'Notify if POS files are not present by 11 PM Saturday'. A 'Create/View Notification' button is at the bottom right, and a 'Synchronize Config' button is at the very bottom.

Notice that the event type for this scenario is a local file search, since the directory to be searched is on the HQ-1 (headquarters) Oracle EBS server. Of course, one POS Update event is defined for each store.

Defining Notifications

In the event that the update files cannot be generated, headquarters needs to be notified via email. This definition is based upon the file not being generated by 11 PM on Saturday. Of course, there should be a notification for each event per store.



The screenshot shows the 'DataZing - Event Configuration Form' with the 'Notifications' tab active. A 'Notification Definition' dialog box is open over it. The dialog has three sections: 'Notification Definition', 'Notification', and 'Notification'. The 'Notification Definition' section has a text field 'Notify if event does not occur by...' with a time picker set to 11:00 AM. Below this is a calendar grid for 'on date(s) of every month:' showing the month of June. The 'Notification' section has fields for 'From Name' (Tina Soles), 'From Email' (tina.soles@we-sell-stuff.com), 'Subject' (Problem with POS Weekly Update), 'To' (administrator_hq@we-sell-stuff.com), and 'Message' (There is a potential problem regarding the POS weekly update. Please notify me when you have a resolution. Thank you.). 'Cancel' and 'OK' buttons are at the bottom.

Defining Actions

Once the files have been generated, they must be FTP'd to each store's server (WSS1 and WSS2). An action to transfer the files via FTP should be defined for Store 1 and for Store 2.

The screenshot shows the 'DataZing - Event Configuration Form' with the 'Actions' tab selected. The 'Event' section at the top contains the following fields:

- Event Type: Local File Search
- Event Seq: 9
- Event Description: Locate files in /home/pos_updates and FTP to Store 1 for updating of all POS terminals
- Directory: /home/pos_updates
- File Pattern: *.upd

Below the 'Event' section is a table for defining actions:

Seq	Action	Description
1	FTP	FTP the POS update files to WSS-1 for updating of the POS terminals

Buttons at the bottom include 'View Events in Sequence', 'Configure/View Action', and 'Synchronize Config'.

Viewing Configured Events

Use the DataZing Event Configuration form to view all events (in order of their sequence) for We-Sell-Stuff, Inc. There are now a total of 10 events, four events for processing transactions from Store 1, four events for processing transactions from Store 2, an event for processing weekly POS updates for Store 1, and an event for processing weekly POS updates for Store 2.

The screenshot shows the 'DataZing - Event Configuration Form' with the 'All Events in Order of Sequence' view. It displays a list of 10 events:

Seq	Event Type User	Event Description
1	Remote File Search (FTP)	Search for 10 AM Batches from Store 1. When found, submit concurrent request
2	Remote File Search (FTP)	Search for 2 PM Batches from Store 1. When found, submit concurrent request
3	Remote File Search (FTP)	Search for 6 PM Batches from Store 1. When found, submit concurrent request
4	Remote File Search (FTP)	Search for 10 PM Batches from Store 1. When found, submit concurrent request
5	Remote File Search (FTP)	Search for 10 AM Batches from Store 2. When found, submit concurrent request
6	Remote File Search (FTP)	Search for 2 PM Batches from Store 2. When found, submit concurrent request
7	Remote File Search (FTP)	Search for 6 PM Batches from Store 2. When found, submit concurrent request
8	Remote File Search (FTP)	Search for 10 PM Batches from Store 2. When found, submit concurrent request
9	Local File Search	Locate files in /home/pos_updates and FTP to Store 1 for updating of all POS te
10	Local File Search	Locate files in /home/pos_updates and FTP to Store 2 for updating of all POS te

An 'OK' button is visible at the bottom right of the window.

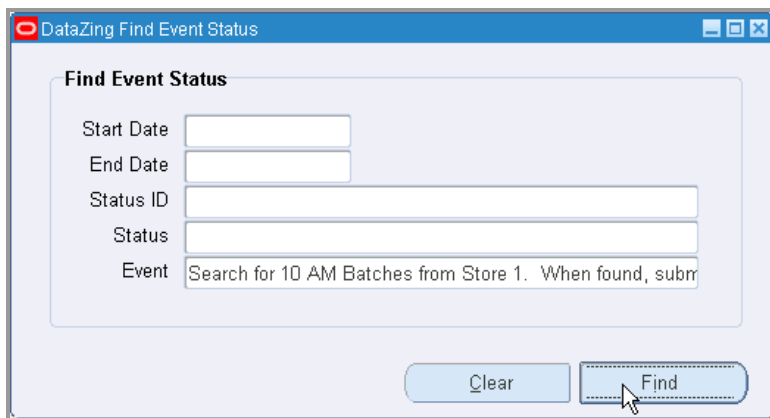
Putting it all together

Once configuration is defined for events, notifications, processing rules, and actions, all that remains is for the actual processing to take place by DataZing, which is comprised of an “engine” that runs continually to perform the processing based upon the configuration.

When an event occurs, processing rules are checked, and if an action and/or notification is to be taken, DataZing performs the action and/or notification. All of this processing takes place automatically, with absolutely NO user intervention required! And, if maintenance needs to be performed, such as the changing of a password for FTP transfers, no additional staff needs to be involved – simply bring up the DataZing Event Configuration form and change information with the Event Configuration Form.

Checking on the Status of Events

With so much automated processing going on “behind the scenes” it would be nice if EBS users could check on the status of the events that DataZing is processing at any given time. This is easily accomplished through the use of the DataZing Event Status form. This form allows a user to query the database based on criteria such as Status ID, Event Name, Status, and Start/End dates.



The screenshot shows a Windows-style application window titled "DataZing Find Event Status". Inside the window, there is a section titled "Find Event Status" containing five input fields: "Start Date", "End Date", "Status ID", "Status", and "Event". The "Event" field contains the text "Search for 10 AM Batches from Store 1. When found, subm". Below these fields are two buttons: "Clear" and "Find". A mouse cursor is pointing at the "Find" button.

For example, to check on the processing of the 10 AM transaction batch from store 1, the user could enter the event name to query for this particular event. Upon clicking “Find”, the user is then presented with the DataZing Event Status form, with a list of the past events matching the criteria are populated.

The screenshot shows a window titled "DataZing Event Status". It contains a table with the following columns: ID, Event Description, Status Date, Status Code, and Status Message. The first row is populated with the following data:

ID	Event Description	Status Date	Status Code	Status Message
19	Search for 10 AM Batches from Store 1. When found,	25-MAR-2010 10:07:11	SUCCESSFUL	Event processing completed successfully.

Below the table are four buttons: "View Notifications", "View Processed Rules", "View Actions", and "Refresh Results".

The form displays information such as the Event ID, Event Description, Status Date, Status Code, and Status Message. In this example, the processing of the 10 AM batch was successful.

From this form, the user can further “drill down” into the details of the event itself, along with the outcome of notifications, processing rules and actions that were specifically associated with this event.

The notification associated with this event was to only take place if the 10 AM batch was not present. Since the 10 AM batch was processed, there are no notifications that took place with this event, as viewed by clicking the View Notifications button.

The screenshot shows a window titled "DataZing (Notification Outcome for Event: Search for 10 AM Batches fro)". It contains a table with the following columns: ID, Notification Description, Date, and Message. The table is currently empty.

ID	Notification Description	Date	Message

An "OK" button is located at the bottom right of the window.

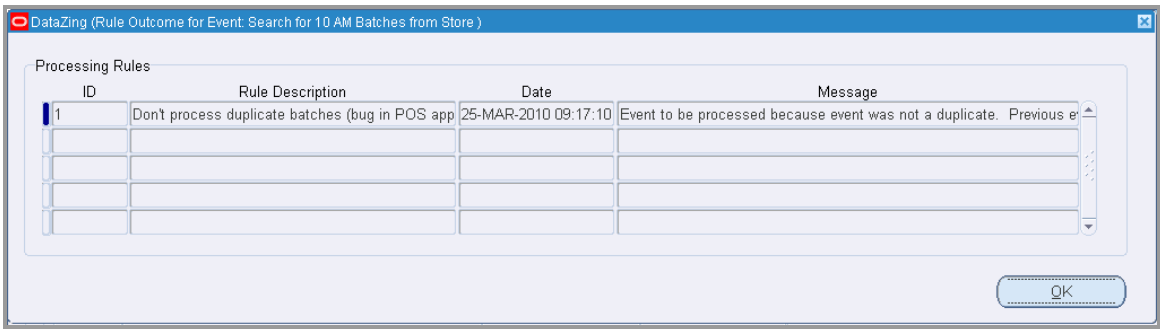
Had the event not taken place by our defined time period of 10:15 AM, the notification would look like this:

The screenshot shows a window titled "DataZing (Notification Outcome for Event: Search for 10 AM Batches fro)". It contains a table with the following columns: ID, Notification Description, Date, and Message. The first row is populated with the following data:

ID	Notification Description	Date	Message
1	Notify administrator if 10 AM batch is not present	25-MAR-2010 10:15:21	Notification sent because event has not occurred by 10:15 AM on day o

An "OK" button is located at the bottom right of the window.

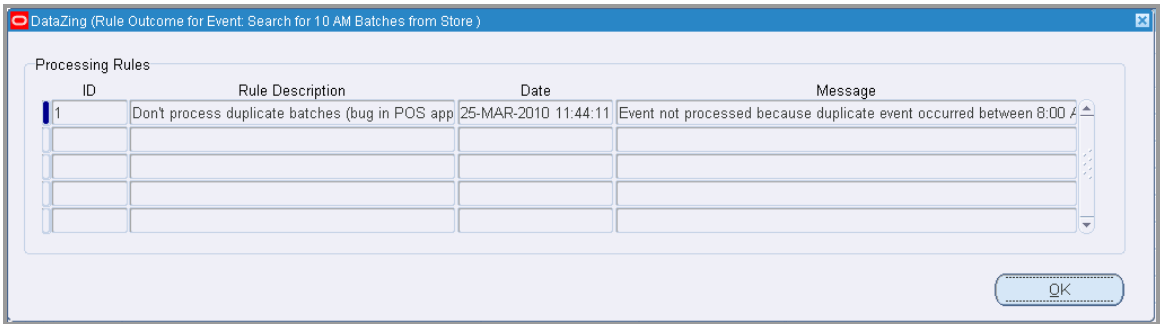
There was a processing rule also configured for this event regarding the processing of duplicate batches. By clicking the View Processed Rules button, we see that the event was processed because no duplicate batches were found.



The screenshot shows a window titled "DataZing (Rule Outcome for Event: Search for 10 AM Batches from Store)". Inside, there is a section labeled "Processing Rules" containing a table with four columns: ID, Rule Description, Date, and Message. The first row shows ID 1, Rule Description "Don't process duplicate batches (bug in POS app", Date "25-MAR-2010 09:17:10", and Message "Event to be processed because event was not a duplicate. Previous e". There are four empty rows below it. An "OK" button is at the bottom right.

ID	Rule Description	Date	Message
1	Don't process duplicate batches (bug in POS app	25-MAR-2010 09:17:10	Event to be processed because event was not a duplicate. Previous e

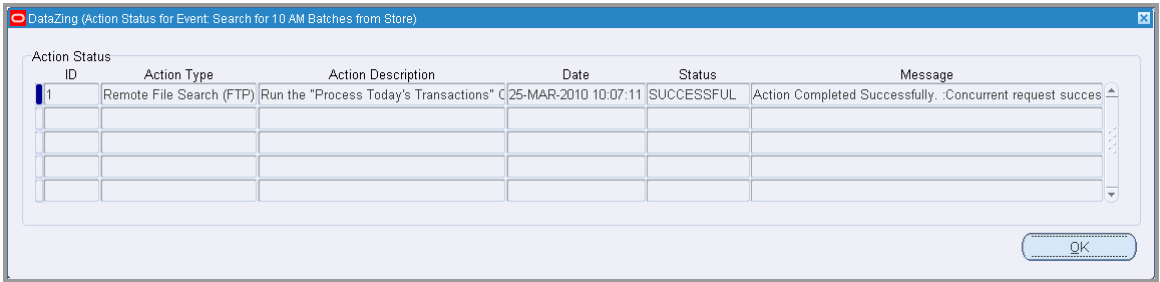
Had there been a duplicate batch, the Processed Rules would look like this:



The screenshot shows a window titled "DataZing (Rule Outcome for Event: Search for 10 AM Batches from Store)". Inside, there is a section labeled "Processing Rules" containing a table with four columns: ID, Rule Description, Date, and Message. The first row shows ID 1, Rule Description "Don't process duplicate batches (bug in POS app", Date "25-MAR-2010 11:44:11", and Message "Event not processed because duplicate event occurred between 8:00 A". There are four empty rows below it. An "OK" button is at the bottom right.

ID	Rule Description	Date	Message
1	Don't process duplicate batches (bug in POS app	25-MAR-2010 11:44:11	Event not processed because duplicate event occurred between 8:00 A

Lastly, we want to ensure that the concurrent request action associated with this event was executed. By clicking the View Actions button, we see that the concurrent request was successfully executed.



The screenshot shows a window titled "DataZing (Action Status for Event: Search for 10 AM Batches from Store)". Inside, there is a section labeled "Action Status" containing a table with six columns: ID, Action Type, Action Description, Date, Status, and Message. The first row shows ID 1, Action Type "Remote File Search (FTP)", Action Description "Run the 'Process Today's Transactions' C", Date "25-MAR-2010 10:07:11", Status "SUCCESSFUL", and Message "Action Completed Successfully. :Concurrent request succes". There are four empty rows below it. An "OK" button is at the bottom right.

ID	Action Type	Action Description	Date	Status	Message
1	Remote File Search (FTP)	Run the "Process Today's Transactions" C	25-MAR-2010 10:07:11	SUCCESSFUL	Action Completed Successfully. :Concurrent request succes

Summary

This paper provides a solution to a number of challenges that face organizations with regards to exchange of data between Oracle EBS and third-party systems. Using a scenario at We-Sell-Stuff, Inc., we saw that DataZing easily handles all of these challenges, providing an organization with a cost-effective solution to remedy these issues.

Challenge	DataZing Solution
Our technical staff is tied up with creating and maintaining scripts and programs to automatically perform simple data interchange functions.	<p>DataZing is completely dedicated to handling the processing of events associated with data exchange. Simply configure an event and let DataZing handle it for you, allowing you to eliminate custom scripts and programs.</p> <p>Your technical staff is no longer needed to maintain these customized processes so that they can make more productive use of their time!</p>
Our employee productivity is not being fully realized, as the staff is constantly interrupted to perform manual transfers of files between third-party applications and Oracle EBS.	DataZing can automatically transfer files, submit concurrent programs, email data, rename a file and more. No more human intervention is required!
Someone needs to understand each of the third-party systems and how they integrate with Oracle EBS in order to effectively exchange data between the systems.	DataZing does not need to integrate into a third-party system – it only needs a directory name and a file pattern to exchange the data between Oracle EBS and the third-party system.
We need something that we can “plug and play” right into our Oracle EBS forms environment, where very little implementation time is involved, and little to no training is required.	Simply install DataZing, configure it, and start the software, all in a matter of a few hours! The DataZing Oracle-based forms are intuitive, making the learning curve insignificant for the user.
We need some way to monitor in real time what is happening with our data exchange processes.	DataZing’s powerful Event Status form allows a user to query for events based upon specific criteria and to view the notifications, rules, and actions associated with each event. Additionally, DataZing’s notification functionality allows the proper staff to be involved in the event of a problem during processing.
Although there are other third-party solutions available which may address my needs, they are complex, do not sit within EBS and are very expensive.	DataZing leverages your present Oracle investment, and is dedicated to the handling of events related to data exchange. Best part is, it all takes place from within EBS. No UNIX admin or custom code required.

Case Studies

Explore how installed customers are using DataZing to save time and money by automating data exchange and event driven-processes from within Oracle EBS.

[Mutual Materials](#) – Leading manufacturer, distributor and retailer of masonry products in the Pacific Northwest wants to upload daily transactions from POS systems to Oracle EBS – automatically. DataZing does that and more – saving 50 hours of work each week by automating a single process.

Product Demonstrations

Experience STR Software's DataZing product live within an Oracle EBS environment. Visit <http://www.strsoftware.com/webcast.htm> and sign up for our next live product demonstration.

About STR Software

DataZing is developed, marketed, and supported by STR Software. Having over 24 years of business-to-business software development experience, STR Software is known for providing "Solutions That Run" with over 1,250 licenses sold worldwide. Headquartered in Richmond, Virginia, STR Software is also known for creating enterprise level automated fax, internet fax, email and printing solutions. STR Software is an Oracle PartnerNetwork Platinum Partner.

Contact STR Software

To learn more about how STR Software can help save your company time and money, please visit us online at www.strsoftware.com or call us at 804-897-1600 x2 (toll free 800-897-7097).

All products and companies herein may be registered or unregistered trademarks of their respective owners and are hereby acknowledged.



1505 Allecingie Parkway · Richmond, VA 23235
office: 800-897-7097 office · fax: 804-897-1638
www.strsoftware.com