

## **The SOA Gateway and EntireX Persistent Messaging**

**This document illustrates how  
EntireX persistent messaging and the  
SOA Gateway can ensure that business IT  
systems can continue to function over planned and  
unplanned outages of critical resources such as ADABAS in  
a standard way with absolutely no loss of data or functionality**

This document is distributed for information purposes only and does not form part of or constitute an agreement with Risaris Ltd. Although Risaris Ltd. uses reasonable efforts to include accurate and up-to-date information in this document, Risaris makes no warranties or representations as to its accuracy. Risaris Ltd. may also make improvements and/or changes to this document at any time without notice. The various approaches outlined in this document are put forward in good faith, but it remains possible that individual results may vary. For that reason and in accordance with standard practice, readers are encouraged to test any materials developed on the basis of this paper before putting them into productive use.

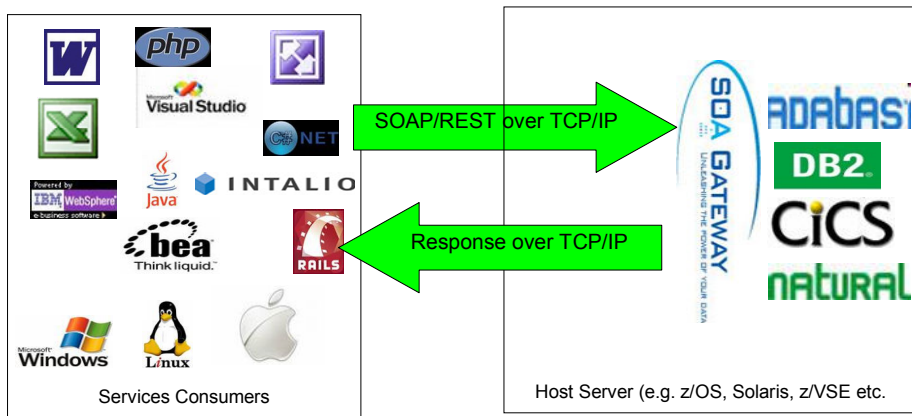
## 1. Overview

All organizations are now so dependent on their IT infrastructures that any outage, whether planned or unplanned, can cause massive inconvenience. However, there are regular points where planned outages are required sometimes nightly, weekly, monthly or even yearly and thus the business applications must be designed to continue operating in as far as possible while specific resources are unavailable.

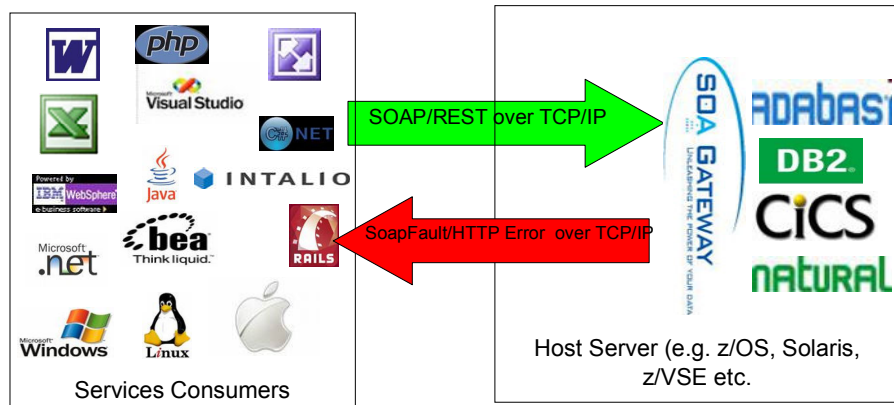
This is not a simple task as enabling continuous operation while resources are unavailable must be designed into an application from the start. The traditional approach to this was very effective but used proprietary interfaces, took months to implement and was extremely costly. This document will illustrate how using the standards based approach offered by the SOA Gateway technology and the message persistence offered by EntireX Communicator, continuous operation can be supported quickly and in a cost effective manner by enabling the project to focus on the business processes required instead of the plumbing.

## 2. Standard Access to Resources

The SOA Gateway currently offers REST and SOAP based access to resources using TCP/IP as per the following diagram:



This provides a standard service to access various resources from multiple technologies over a TCP/IP connection. This is all fine until such time as a resource such as ADABAS is not available for some reason. Due to the non persistent nature of TCP/IP, the only option available is to fail the request with a SOAP fault as illustrated in the next diagram:

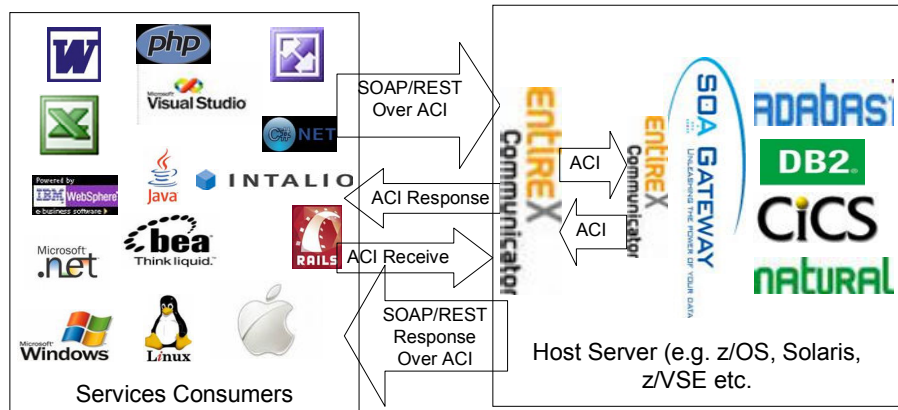


When such an event occurs, it means that the consuming applications can no longer continue as their request has failed. For example, if the application is dependant on a record being added to or delete from the ADABAS database, it cannot continue as it knows that this will not be done.

This level of service is unacceptable today where more and more companies are moving to 24\*7 levels of service and outages for organizations can literally cost millions of dollars for every minute they are down. The next section illustrates how through the use of EntireX Communicator as a transport, this problem can be resolved.

### 3. Standards Based Access with Persistence

Once the service is defined, the client applications know the format of the request and response messages. It has been illustrated how TCP/IP access works well when resources are available but offers no comfort when a resource is unavailable. The SOA Gateway offers the capability to send the same messages using EntireX as the transport as per the following diagram:

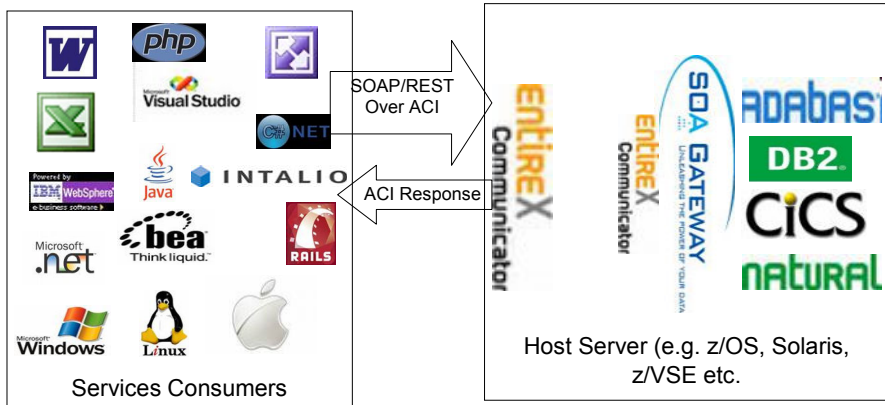


Under normal circumstances when the resources are available, the request is passed directly to the SOA Gateway which then interfaces with the appropriate resource. For example, a record will be added to the ADABAS database. If a request is made to return some data, this data will be made available immediately via the response.

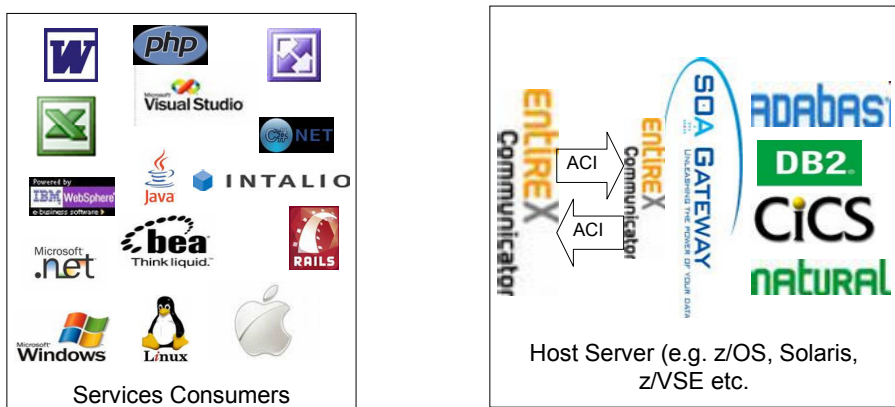
There are a number of reasons why a resource may become unavailable in an installation. The following are just some examples:

- An installation may have a batch window where they do not want interference from external processes.
- An installation may want to avoid external processes impacting on their critical resources during online hours.
- Software maintenance updates require that the resource is unavailable until the maintenance window is completed.
- If the resource suffers a software failure, it will become unavailable inadvertently without warning.

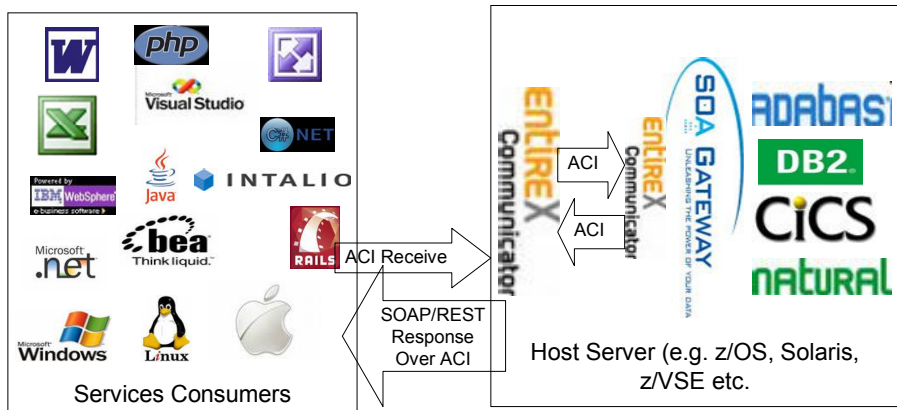
In all the above scenarios, it would be preferable that other work, not directly impacted by the unavailability of the resource, can continue. Using EntireX persistent messaging this is possible as per the following illustration:



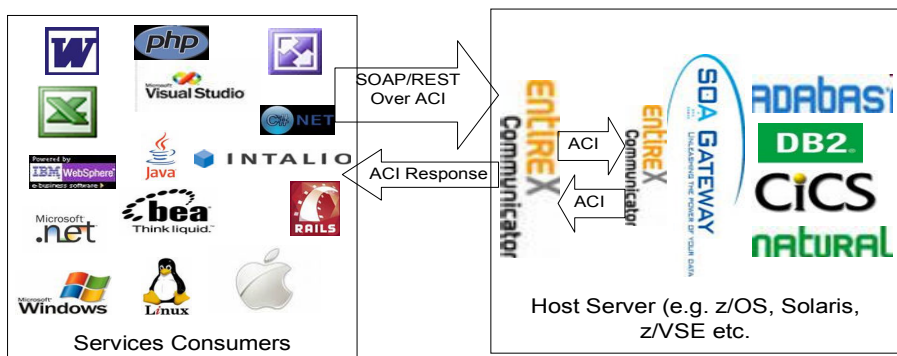
While the resource is unavailable, such as an ADBAS database or File, EntireX accepts the requests from the clients and provides an ACI response to indicate that the message will be delivered. The process can then continue as it knows that while the database update or application invocation may not occur immediately, EntireX will ensure that it does occur. When the resource or resources are available again, EntireX delivers the messages to the SOA Gateway and they are applied as normal as per the following:



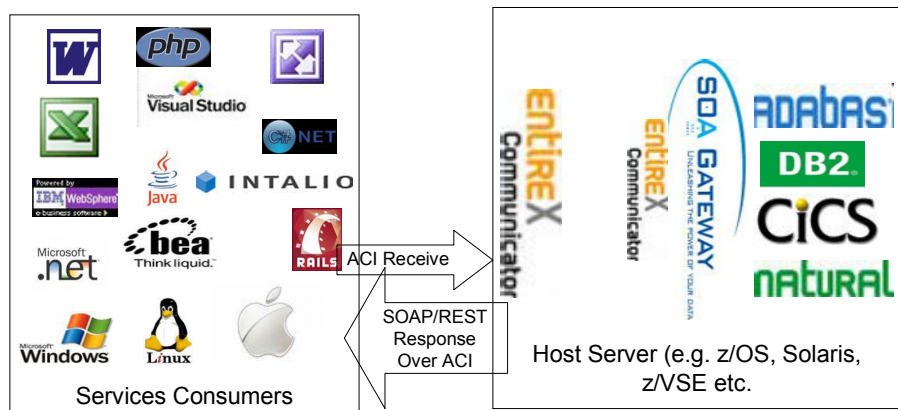
Obviously while this is ongoing, EntireX will continue to receive messages from the other business process so that business continues and ultimately, the SOA Gateway will process all the persisted messages and catch up to processing the latest messages in 'real time'. At the same time, if data is expected to be delivered from the SOA Gateway, it can then be delivered and made available to the application for processing as follows:



This can also be helpful in scenarios where the application processing the returned messages is unavailable for some reason as follows:



The data can be requested and processed by the SOA Gateway which will in turn pass this to EntireX Communicator which can confirm receipt and enable the SOA Gateway to continue. As there is no client available to receive the data, EntireX Communicator will persist the data and make it available when the client is available to receive the data as follows:



The above clearly illustrates how using EntireX as a transport mechanism provides flexibility for core resources to be unavailable in a planned or unplanned way and to enable business to continue with no loss of data. In addition, it can provide the flexibility to have the data available to external process to be picked up when the external process is ready to receive the data.

#### 4. Usage Scenarios

There are a number of scenarios where this transport can be helpful.

1. When core databases resources must be made completely unavailable for whatever reason, EntireX can be used to persist additions, deletes or updates for a database like ADABAS or DB2 until such time as the database is available again. At that time the, SOA Gateway will apply the updates to the database and EntireX can then remove all knowledge of the messages. Requests to return data may also be persisted such that when the database returns, the SOA gateway can process the request and provide the results to EntireX which can persist the result until they are picked up by the appropriate processing application.
2. If a database is only required to be closed for updates, it would be possible to get immediate responses to read only requests using the TCP/IP access while pushing updates via the EntireX route to be applied to the database when the read only

session was completed.

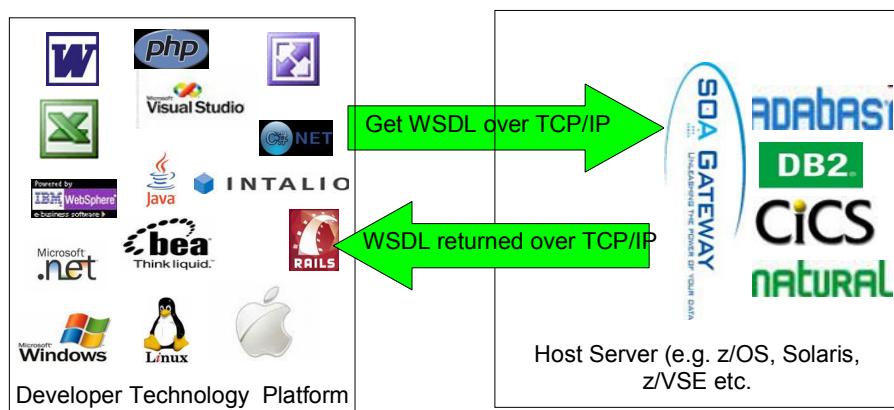
3. If only parts of a database need to be unavailable in the above scenarios, this solution will work equally well.
4. Where application environments like CICS or Natural need to be made unavailable for some time, again the calls to these environments may be persisted by EntireX until they are available again. At that time the SOA Gateway will invoke the appropriate business logic based on the messages that are retrieved from EntireX. If output is to be made available from these calls, this will again be provided to EntireX which will retain the response until the appropriate processing application is available to pick this up.
5. Again, this scenario may be applied for small number of applications or a single application that need to be rendered unavailable and will work equally well.
6. If an organization wishes to protect itself from hardware failure or provide windows for hardware replacement, the EntireX Communicator server can reside off the platform where the SOA Gateway resides. In this way, if the hardware where the SOA Gateway fails, has problems or needs to be replaced, EntireX Communicator will persist the messages until such time as the hardware and the SOA Gateway are again available to process the messages.

## 5. Error Processing

Each SOA Gateway service that is defined will have an error queue defined which will be represented by an EntireX service. When errors occur processing a message received from EntireX either to access a database or call an application, the error will be written to this queue. Part of the processing logic of the business process for this application will be to regularly check this queue for errors so that these can be reviewed and corrected as appropriate.

## 6. Development Requirements

This solution uses standards to make the integration of services defined by the SOA Gateway via EntireX messaging as simple as possible. Consider a developer who wishes to create an application which makes use of a SOA Gateway service for a database. Their first task will be to import the WSDL into their environment from the SOA Gateway as per the following illustration:



The WSDL will tell the user in general terms:

1. The methods that can be called for the database such as add, delete, update, list, select or get.
2. The format of the messages and responses for each method.
3. The transports that may be used to issue those messages. In this case, the user will have two choices:

- a. TCP/IP for which an endpoint will be provided containing the host, port and URI name where the SOAP message may be sent and from which a response will be received.
  - i. Note that for TCP/IP, all methods are invoked using SOAP Request/Response protocols.
- b. EntireX Broker for which a service will be provided containing the broker name, class, service name and service to be used by the method.
  - i. For Broker transport, the methods will be broken down as follows:
    1. Add, update and delete will be request only. The reason for this is that when EntireX has accepted the request, it will ensure that the SOA Gateway adds this data so no response is required.
    2. List, Get and the Select\* methods will have a Request only capability to send the request and a response only capability to pick up the response. This means that different logic can be created to create the request for data while separate associated logic can pick up the response when it is ready to process it.

The developer constructs their application as they would to use any other TCP/IP based Web Service to create the request messages. When the developer indicates that they wish to use EntireX as the transport for one or more methods, the SOA Gateway will deliver a SOA Gateway transport implementation for their environment. This will initially be delivered in Java or .Net format and will take the SOAP request and issue the appropriate EntireX Communicator call using the EntireX Communicator Stub (Subject to license). This will send the SOAP request to EntireX Communicator.

If the resource is available and active, the SOA Gateway will receive the SOAP message off the queue, process it and, once processed, will commit the message on EntireX so that it can be deleted.

If a response is required to the message, the SOAP response will be constructed according to the format published in the WSDL. This will then be sent to the output queue which will also be an EntireX Service.

This SOA Gateway transport implementation will receive the SOAP response from EntireX Communicator and make it available to the Java or .net application for processing.

Apart from this one call to send or receive the SOAP request or response, the processing of the SOAP request and response will be identical to that required for the TCP/IP transport. This processing is second nature to programmers developing in the .net and Java environments.

It should be noted that this capability is also available using REST over EntireX Communicator. As the REST standard is a lot less complex than the SOAP standard, these requests may be built by hand (as they must be for the TCP/IP transport) with the only difference being again that the EntireX transport capability is used rather than the TCP/IP capability.

## **7. Summary**

This document has shown how the usage of standards with the EntireX Communicator persistent messaging capability can help an organization achieve 24\*7 levels of service while focusing on the deliverables to benefit the business rather than the architecture. This is achieved with the following benefits:

- Development to access the service is identical to dealing with any SOAP or REST based Web Service and thus developers need no additional training. This is all handled under the covers by the technology.
- Core database or application assets can be made 100% unavailable while systems that use those systems can continue to operate unaffected.
- The flexibility to split read only and update access provides a 'best of both worlds' approach when databases or applications must be unavailable for update only.
- Applications using the data also benefit in that they can pick up the data automatically when they are ready to process it while EntireX will ensure that it is available until such time as it is read by the process.
- The solution can also be configured to protect against hardware failure through the strategic placement of the SOA Gateway and EntireX Communicator servers.
- Once a service is defined once, it can be reused again and again by other applications or technologies no matter what platform they are running on or what language they are written in.
- Due to the standards based approach adopted by the solution, it can be implemented and delivered in a fraction of the time required for traditional implementations.