# Introduction - Putting it all Together …

We’re going to work within the confines of a SOA developed by **Omnicorp Corporation** HL7 v2.x schema

**Purchase-Totaller** for the GIORP-5000 o Given a customer’s total GIORP purchase as well as the region they live in in Canada, you will develop a service to determine and return the sub-total, provincial sales tax – PST (if any), harmonized sales tax – HST (if any) and the goods and services tax – GST (if any) as well as the purchase grand total.

# Assignment Requirements

# Communication, Development Languages and OS’s

The underlying communication scheme that you will use used to support the SOA Registry and SOA messaging protocol is the stream ***socket.***

Developed in C# running in Windows 7 OS

I have developed and provided the SOA Registry for you to use. The runtime (and necessary supporting files / directory structure) are found in this ZIP file associated with this assignment. The SOA Registry was written in Java and requires you to customize a few files before trying it out in your development – **please see the README file in the extracted “SOA” root directory**.

# Service Design and Runtime Logging

When designing your different SOA services, it is important that you follow best practices and ensure that your service(s) are and have :

* Modular design – to maximize re-use of code and cohesive design
* Full class/file, method/function and inline commenting
* Full error handling / input validation capabilities when talking to the SOA Registry and other calling applications
  + You will need to consider the purpose of the service and what type of error handling and validation you need to provide
* Retrieve their runtime settings from a configuration file at start-up
  + Much in the same way as the SOA Registry from was given to you
  + This will allow you to be able to develop your SOA more easily and be able to more fully test it!

It is also important that each of your services log information during their execution. Your team will be judged and marked based upon your source code as well as the various run-time logs produced by your services on demo day. Please ensure that all log entries are date and time stamped and that the following events and information is logged :

* Your team name as well as the service’s tagName and serviceName
* Messages going out to SOA Registry as well as the responses that come back
* Incoming service requests (Execute Service message) contents including
  + The calling team and their ID
  + The serviceName, arguments and values
* The response that your service creates to send back to the requesting team
  + Any and all error conditions and exception conditions (including socket errors)
  + See example Service Log in Appendix A of this assignment

Keep in mind that the team name you choose for your 4 services must all be the same. As well this team name should be branded on your consumer application….

# Consumer Design and Runtime Logging

You also need to develop a Windows GUI based program (in C#) in order to demonstrate the calling of the SOA Registry and other services. This application needs to be able to prompt the user for required information as necessary. For example if the user decides to call the CAR-LOAN service, your SOA user application must prompt the user for the arguments as specified in the SOA Registry’s response to the Query Service Message. As an added feature, the user application can also do some input validation on these argument values given that you were told the data-type of the arguments as well.

From the user perspective, your client application should work is as follows :

* The first screen should ask the user which of the 4 services they wish to execute – and the user can select
  + When a selection is made, your application puts out a call to the SOA-Registry looking for a service to perform that task
* The next screen is presented to the user after your application has asked the SOA-Registry for information about the selected service. This is the screen where your client application needs to parse and understand the information coming back about the service to call in terms of the service’s name, parameter names and data-types and optionality. o This screen should display the team name that has published the service about to be executed, the service name and the service description
  + This screen will allow the user to enter their input information and select an EXECUTE function
* Once the user has EXECUTED, your client application may wish to run simple data-type validation on the user inputs before calling the actual service. At some point, your application calls the service that you were given with the user’s input information and awaits a result / response
  + When the client application receives the response, this input screen will be updated with the service’s response / results
  + The user will be able to select a DONE option which takes them back to the first screen of your application once again

Just as in the case of the individual services, the user application must produce a log file as well – it is part of the mark you will receive on the application. See the example User-App Log in Appendix B of this assignment.

# Other Considerations

Given that you are developing 4 different services in 4 different languages, you might actually want to use some of your PAD knowledge to define a generic socket-communication interface with OS and language specific code behind it. This is an example of an *Abstraction Layer* and as far as having a SOA is concerned, this type of design abstraction is the basis of SOA Interface policies.

As well, keep in mind that the SOA Registry can be configured to assign time-outs to your team and service registration. So it may be possible that the Registry will drop your team and service information at some point. You will need to develop a strategy using the existing SOA Registry commands to detect this situation and properly react to it.

# When is this thing due?

* As mentioned in class, this assignment is officially due Sunday November 30, 2014 by 11:59pm in the eConestoga Dropbox.
* You will need to submit the source code, any makefiles – or build scripts that are needed to recompile and regenerate your executable(s) as well as any configuration files that your services may need.
* Also as mentioned in class, you and your team will end up demoing your services and client on December 1, 2014 during the last SOA class period of the term – at the same time as every other team is demoing
  + It might not be a bad idea with make sure that your team’s services can co-exist and *play nicely* with other services from other teams

# Service Logic (Purchase-Totaller for the GIORP-5000)

* The tagName for this service must be GIORP-TOTAL, you choose your service’s security level from a value of 1 (LOW), 2 or 3 (HIGH).
* You may name your specific service however you like (e.g. giorpTotaller or totalPurchase) o The service needs to take 2 mandatory arguments – these arguments can have any name or argument order but the argument values need to be passed into this service
  + One argument needs to capture the Province or Territory that the purchase is being made in – allowable argument values are shown in the table below
* The other argument needs to be the total value (in dollars and cents) of the purchase being made
  + You may add other arguments if you like
* The idea behind this service is to calculate and return the purchase sub-total, various tax additions and grand-total for the purchase given the Province/Territory it is being made in – the following table shows the Provincial / HST and GST rates for the various regions
* In regions where an HST rate is specified – there is no additional PST and GST calculated (and vice versa)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Allowed  Region Code | Code Meaning | PST Rate | HST Rate | GST Rate |
| NL | Newfoundland | -- | 13% | -- |
| NS | Nova Scotia | -- | 15% | -- |
| NB | New Brunswick | -- | 13% | -- |
| PE | Prince Edward Island | 10% \* | -- | 5% |
| QC | Quebec | 9.5% \*\* | -- | 5% |
| ON | Ontario | -- | 13% | -- |
| MB | Manitoba | 7% | -- | 5% |
| SK | Saskatchewan | 5% | -- | 5% |
| AB | Alberta | 0% | -- | 5% |
| BC | British Columbia | -- | 12% | -- |
| YT | Yukon Territories | 0% | -- | 5% |
| NT | Northwest Territories | 0% | -- | 5% |
| NU | Nunavut | 0% | -- | 5% |
| \*In PEI, the PST is calculated on the purchase price + GST.   eg. If you purchase something for $100.00, then the GST is $5.00 and the PST is $10.50 (=10% of $105.00)  \*\* In Quebec, the PST is calculated on the purchase price + GST. e.g. if you purchase something for $100.00, then the GST is $5.00 and the PST is $9.98 (=9.5% of $105.00) | | | | |

* This service must return 5 different values – the names of the return values can be whatever you like and specified in any order, but must capture the following information
* Sub-total amount – which is the value that was passed into the service as the purchase amount
* PST amount – the value to be added to the purchase amount given the information in the above table
* HST amount – the value to be added to the purchase amount given the information in the above table
* GST amount – the value to be added to the purchase amount given the information in the above table
* Total purchase amount – the sub-total amount plus the PST amount plus the HST amount plus the GST amount

# SOA Messaging Protocol and Scheme

## SOA Messaging Scheme

As you know from Lesson 1, a SOA is really based upon a clear, well defined set of rules, messaging schemes and protocols. At the heart of any SOA is the Registry where information about these policies, the services, etc. is stored and enforced. I have developed the SOA-Registry that you must use – it adheres to the schema and protocols discussed in this assignment. (You can download it from the course Assignment space)

As mentioned, our SOA is based upon the HL7 v2.x messaging schema (please see page 12 of this assignment for a detailed description of the messaging protocol.

Below you will find a summary of the 6 different SOA processing messages and the 2 variations of allowed responses to each of the messages. Elements notes in ***bold italics*** are pieces of information that you need to provide when sending your messages. Examples of these messages are included in the sample client directory of the given SOA Registry zip file.

1. ***Register Team* Message**

|  |  |  |
| --- | --- | --- |
| **Description:** | | |
|  | Before you can begin to do anything with the SOA Registry or its services, you need to register your team. This is done by sending a REG-TEAM message to the registry with your team name. If the registration succeeds, you will receive a ***teamID*** from the Registry and an expiration time as well (for your Registry license). The registry will also assign your team a security level (which it doesn’t tell you about) which may prevent you from executing some services …  | |
| **Message Format:** | | |
|  | DRC|REG-TEAM||| INF|***<team name>***||| | |
| **Response Format:** | | |
|  | The SOA Registry will respond to your message with either an OK or NOT-OK status with the information specified below. | |
|  |  | PASS: SOA|OK|<teamID>|<expiration>|| FAIL: SOA|NOT-OK|errorCode|errorMessage|| |

1. ***Unregister Team* Message**

|  |  |  |
| --- | --- | --- |
| **Description:** | | |
|  | If for some reason you want to remove your team (and its services) from being published in the Registry – then you send an UNREG-TEAM message | |
| **Message Format:** | | |
|  | DRC|UNREG-TEAM***|<team name>***|***<teamID>*** | |
| **Response Format:** | | |
|  | The SOA Registry will respond to your message with either an OK or NOT-OK status with the information specified below. | |
|  |  | PASS: SOA|OK|<teamID>|<expiration>|| FAIL: SOA|NOT-OK|errorCode|errorMessage|| |

1. ***Query Team* Message**

|  |  |  |
| --- | --- | --- |
| **Description:** | | |
|  | When a service you published receives a request to perform a task (i.e. your service receives an *Execute Service Message*), you need to verify that the calling team is allowed to perform that service. The response from this message indicates if the calling team is still valid (and hasn’t expired) and if they have the necessary security level to call your service. | |
| **Message Format:** | | |
|  | DRC|QUERY-TEAM***|<team name>***|***<teamID>***|  INF|***<team name>***|***<teamID>***|***<service tag name>***| | |
|  | **NOTE:** The ***<team name>*** and ***<teamID>*** in the DRC segment represent your team’s name and ID.  The ***<team name>*** and ***<teamID>*** in the INF segment represent the team that you are asking about. The ***<service tag name>*** in the INF segment is the tagName of **your service** that this team wishes to execute. | |
| **Response Format:** | | |
|  | The SOA Registry will respond to your message with either an OK or NOT-OK status with the information specified below. | |
|  |  | PASS: SOA|OK|<teamID>|<expiration>|| FAIL: SOA|NOT-OK|errorCode|errorMessage|| |

1. ***Publish Service* Message**

|  |  |  |
| --- | --- | --- |
| **Description:** | | |
|  | Call the SOA Registry with this message in order to publish a service for your team so that other teams may begin to call it. See commented example message for a description of this message’s segments and elements. | |
| **Message Format:** | | |
|  | DRC|PUB-SERVICE***|<team name>***|***<teamID>***|  SRV|***<tag name>***|***<service name>***|***<security level>***|***<num args>***|***<num responses>***|***<description>***|  ARG|***<arg position>***|***<arg name>***|***<arg data type>***|***[mandatory | optional]***||  . . .  RSP|***<resp position>***|***<resp name>***|***<resp data type>***||  . . .  MCH|***<published server IP>***|***<published port>***| | |
|  | **NOTE:** The ***<team name>*** and ***<teamID>*** in the DRC segment represent your team’s name and ID.  The ***<team name>*** and ***<teamID>*** in the INF segment represent the team that you are asking about. The ***<service tag name>*** in the INF segment is the tagName of **your service** that this team wishes to execute. | |
| **Response Format:** | | |
|  | The SOA Registry will respond to your message with either an OK or NOT-OK status with the information specified below. | |
|  |  | PASS: SOA|OK|<teamID>|<expiration>|| FAIL: SOA|NOT-OK|errorCode|errorMessage|| |
| Note: | | |
|  | 1. Only (somewhat) primitive data-types are allowed in the ARG and RSP segments. The allowed data-types are:  * char * short * int * long * float * double * string  1. Select port to publish in a range greater than 2000 | |

1. ***Query Service* Message**

|  |  |  |
| --- | --- | --- |
| **Description:** | | |
|  | If you need to call a service to perform some work for you – you send the SOA Registry this message asking for information about a published service capable of doing that work for you. | |
| **Message Format:** | | |
|  | DRC|QUERY-SERVICE***|<team name>***|***<teamID>***|  SRV|***<tag name>***|||||| | |
| **Response Format:** | | |
|  | The SOA Registry will respond to your message with either an OK or NOT-OK status with the information specified below. In the case of an OK response, you will receive the information about the service that you are allowed to call – so you will need to parse this response and use the information in it to construct your *Execute Service Message*. | |
|  |  | PASS: SOA|OK|||<num segments>|  SRV|<team name>|<service name>||<num args>|<num responses>|<description>|  ARG|<arg position>|<arg name>|<arg data type>|[mandatory | optional]||  . . .  RSP|<resp position>|<resp name>|<resp data type>||  . . .  MCH|<published server IP>|<published port>|  FAIL: SOA|NOT-OK|errorCode|errorMessage|| |
| **Note:** | | |
|  | 1. The ***<num segments>*** value represents the number of message segments that follow the SOA|OK 2. Please note that the second field in the SRV tag of the SOA|OK response contains the team name of the team that published the service. | |

1. ***Execute Service* Message**

|  |  |  |
| --- | --- | --- |
| **Description:** | | |
|  | After calling the SOA Registry and sending a *Query Service Message*, you need to parse out the response (assuming it was successful) and construct the following message which you send to the service’s machine on its designated port. Remember that when your service receives this message from another team – you need to call the SOA Registry with the *Query Team Message* in order to see if that team is allowed to call your service or not. | |
| **Message Format:** | | |
|  | DRC|EXEC-SERVICE***|<team name>***|***<teamID>***|  SRV||***<service name>***||***<num args>***|||  ARG|***<arg position>***|***<arg name>***|***<arg data type>***||***<arg value>***|  . . . | |
| **Response Format:** | | |
|  | All of your published services need to adhere to the SOA messaging scheme and respond to the calling team with the following messages | |
|  |  | PASS: PUB|OK|||<num segments>|  RSP|<resp position>|<resp name>|<resp data type>|<resp value>|  . . .  FAIL: PUB|NOT-OK|errorCode|errorMessage|| |

# SOA Messaging Protocol

The HL7 v2.x messaging scheme uses sockets to communicate between different end points (different computer systems) within a medical facility. The HL7 scheme is based upon the idea of message passing – where each message as a single purpose. In this scheme, a message is comprised of multiple segments. Each segment within the message gives information about the purpose of the message. Each segment is comprised of multiple fields – and the first field in any segment indicates the purpose of that segment. A general HL7 v2.x message looks like this:

<BOM>[segment]<EOS>[segment]<EOS> . . [segment]<EOS><EOM>

where

<BOM> is a special character used to mark and indicate the start of a new message

<EOS> is a special character used to mark and indicate the end of a message segment

<EOM> is a special character used to mark and indicate the end of a message

The placement and existence of the special message framing characters is the responsibility of the vendor using / implementing the schema (in this case – you and your team).

In our SOA messaging scheme the following ASCII character codes will be used for framing the messages :

<BOM> is represented by ASCII 11

<EOS> is represented by ASCII 13

<EOM> is represented by ASCII 28

So when you are building a message to send to either the SOA-Registry or a published service – you will need to construct the message and frame it with these special characters. One very special note worth mentioning is that in practice, the <EOM> framing character is followed by a carriage return on the socket. The SOA-Registry ensures that the <EOM> is followed by a carriage return.

If we re-examine one of the example Register Team messages (for team Blotto) as a stream of characters coming across a socket, you will see that the message :

DRC|REG-TEAM|||

INF|Blotto|||

is really transmitted as :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [11] | [68] | [82] | [67] | [124] | [82] | [69] | [71] | [45] | [84] | [69] | [65] | [77] | [124] | [124] | [124] | [13] | [73] | [78] | [70] | [124] | [66] | [108] | [111] | [116] | [116] | [111] | [124] | [124] | [124] | [13] | [28] | [13] |
| <BOM> | D | R | C | | | R | E | G | - | T | E | A | M | | | | | | | <EOS> | I | N | F | | | B | l | o | t | t | o | | | | | | | <EOS> | <EOM> | <EOS> |

where [###] represents the ASCII character codes … Notice the characters highlighted in yellow … does their placement and ASCII character code purpose give you any ideas about how to communicate (read and write) with the socket ?? The character highlighted in green marks the <BOM> and the character highlighted in red marks the <EOM>

# Appendix A - The Service Run-Time Log

2012-09-03 17:30:00 =======================================================

2012-09-03 17:30:00 Team : Blotto (Dan K, Larry D)

2012-09-03 17:30:01 Tag-Name: CAR-LOAN

2012-09-03 17:30:01 Service : myCarLoanCalculator

2012-09-03 17:30:02 =======================================================

2012-09-03 17:30:05 ---

2012-09-03 17:30:05 Calling SOA-Registry with message :

2012-09-03 17:30:05 >> DRC|REG-TEAM|||

2012-09-03 17:30:05 >> INF|Blotto|||

2012-09-03 17:30:08 >> Response from SOA-Registry :

2012-09-03 17:30:08 >> SOA|OK|1180|18:00:05||

2012-09-03 17:30:12 ---

2012-09-03 17:30:12 Calling SOA-Registry with message :

2012-09-03 17:30:12 >> DRC|PUB-SERVICE***|***Blotto|1180|

2012-09-03 17:30:12 >> SRV|CAR-LOAN|myCarLoanCalculator|3|2|3|Service to calculate loan payments|

2012-09-03 17:30:12 >> ARG|1|principalCarAmount|double|mandatory||

2012-09-03 17:30:12 >> ARG|2|interestRate|double|mandatory||

2012-09-03 17:30:12 >> RSP|1|payment36Month|double||

2012-09-03 17:30:12 >> RSP|2|payment48Month|double||

2012-09-03 17:30:12 >> RSP|3|payment60Month|double||

2012-09-03 17:30:12 >> MCH|142.112.50.103|50002|

2012-09-03 17:30:14 >> Response from SOA-Registry :

2012-09-03 17:30:14 >> SOA|OK||||

2012-09-03 17:30:30 ---

2012-09-03 17:30:30 Receiving service request :

2012-09-03 17:30:30 >> DRC|EXEC-SERVICE***|***Shcarpo|1183|

2012-09-03 17:30:30 >> SRV||myCarLoanCalculator||2|||

2012-09-03 17:30:30 >> ARG|1|principalCarAmount|double||15230.56|

2012-09-03 17:30:30 >> ARG|2|interestRate|double||3.75|

2012-09-03 17:30:05 ---

2012-09-03 17:30:35 Calling SOA-Registry with message :

2012-09-03 17:30:35 >> DRC|QUERY-TEAM|Blotto|1180|

2012-09-03 17:30:35 >> INF|Shcarpo|1183|CAR-LOAN|

2012-09-03 17:30:35 >> Response from SOA-Registry :

2012-09-03 17:30:35 >> SOA|NOT-OK|-5|Insufficient Security Level||

2012-09-03 17:30:36 ---

2012-09-03 17:30:36 Responding to service request :

2012-09-03 17:30:36 >> PUB|NOT-OK|-1|Sorry – your team has insufficient permissions to run this service||

# Appendix B - The Consumer Run-Time Log

2012-09-03 17:30:00 =======================================================

2012-09-03 17:30:00 -- USER APP LOG --

2012-09-03 17:30:00 Team : Shcarpo (Sid D, Larry S)

2012-09-03 17:30:02 =======================================================

2012-09-03 17:30:05 ---

2012-09-03 17:30:05 Calling SOA-Registry with message :

2012-09-03 17:30:05 >> DRC|REG-TEAM|||

2012-09-03 17:30:05 >> INF|Shcarpo|||

2012-09-03 17:30:08 >> Response from SOA-Registry :

2012-09-03 17:30:08 >> SOA|OK|1183|18:00:05||

2012-09-03 17:30:12 ---

2012-09-03 17:30:12 Calling SOA-Registry with message :

2012-09-03 17:30:12 >> DRC|QUERY-SERVICE***|***Shcarpo|1183|

2012-09-03 17:30:12 >> SRV|CAR-LOAN|||||

2012-09-03 17:30:14 >> Response from SOA-Registry :

2012-09-03 17:30:14 >> SOA|OK|||7|

2012-09-03 17:30:14 >> SRV|Blotto|myCarLoanCalculator||2|3| Service to calculate loan payments|

2012-09-03 17:30:14 >> ARG|1|principalCarAmount|double|mandatory||

2012-09-03 17:30:14 >> ARG|2|interestRate|double|mandatory||

2012-09-03 17:30:14 >> RSP|1|payment36Month|double||

2012-09-03 17:30:14 >> RSP|2|payment48Month|double||

2012-09-03 17:30:14 >> RSP|3|payment60Month|double||

2012-09-03 17:30:14 >> MCH|142.112.50.103|50002|

2012-09-03 17:30:30 ---

2012-09-03 17:30:30 Sending service request to IP 142.112.50.103, PORT 50002 :

2012-09-03 17:30:30 >> DRC|EXEC-SERVICE***|***Shcarpo|1183|

2012-09-03 17:30:30 >> SRV||myCarLoanCalculator||2|||

2012-09-03 17:30:30 >> ARG|1|principalCarAmount|double||15230.56|

2012-09-03 17:30:30 >> ARG|2|interestRate|double||3.75|

2012-09-03 17:30:36 >> Response from Published Service :

2012-09-03 17:30:36 >> PUB|NOT-OK|-1|Sorry – your team has insufficient permissions to run this service||

2012-09-03 18:05:00 ---

2012-09-03 18:05:00 Calling SOA-Registry with message :

2012-09-03 18:05:00 >> DRC|QUERY-SERVICE***|***Shcarpo|1183|

2012-09-03 18:05:00 >> SRV|POSTAL|||||

2012-09-03 18:05:03 >> Response from SOA-Registry :

2012-09-03 18:05:03 >> SOA|NOT-OK|-1|Team License Expired||

# Appendix C - SOA-Registry Error Codes and Messages

Depending on how you call the SOA-Registry and with which SOA message *payload*, you may find that the registry comes back at you with a response.

SOA|NOT-OK|errorCode|errorMessage|

In order for you to be able to handle and respond to the various errors – you need to know the possibilities of what might be reported. Below is a table of error codes and messages that may come back to you …

|  |  |  |
| --- | --- | --- |
| Error Code | Error Message | In response to |
| -1 | * Message doesn't contain EOM marker * Invalid segment directive found (<segDirective>) * SOA command <whatever you sent> - UNKNOWN * DRC directive has no embedded SOA command * DRC directive not in first message segment * INF directive not in second message segment * SRV directive not in second message segment | Incorrectly formatted incoming message or unknown SOA command |
| -2 | * DRC/REG-TEAM segment not according to Spec. * DRC/UNREG-TEAM segment not according to Spec. * DRC/QUERY-TEAM segment not according to Spec. * DRC/QUERY-SERVICE segment not according to Spec. * DRC/PUB-SERVICE segment not according to Spec. * EXEC-SERVICE command not processed by SOA-Registry. * INF segment not according to Spec. * INF segment not according to Spec (calling teamName is BLANK). * INF segment not according to Spec (requested tagName is BLANK). * SRV segment not according to Spec (service tagName is BLANK). * SRV segment not according to Spec (service name is BLANK). * SRV segment not according to Spec (security level is BLANK). * SRV segment not according to Spec (numArgs is BLANK). * SRV segment not according to Spec (numResps is BLANK). * SRV segment not according to Spec (calling tagName is BLANK). * SRV segment not according to Spec (All fields after <tagName> must be BLANK). * Tagname (<tagName>) is not valid * ServiceName (<serviceName>) contains invalid characters * Security Level (<securityLevel>) contains invalid value * Number of Arguments (<numArgs>) must be greater than or equal to zero * Number of Responses (<numResp>) must be greater than or equal to one * Service Description contains invalid characters * Service Description too long (needs to be less than 200 characters * ARG Segment #<argNum> - ArgPosition (<whichPos>) is not valid - expected <expectedPos> * ARG Segment #<argNum> - ArgName (<argName>) contains invalid characters * ARG Segment #<argNum> - ArgDatatype (<whichDataType>) is not valid * ARG Segment #<argNum> - ArgMandatoryOptional (<mand-option-value>) is not valid | Incorrectly formed *segment* within message |
| -3 | * • Illegal teamName in INF segment. | Invalid content within the *fields* of the message segment(s) |
| -4 | * No team '<teamName>' (ID : <teamID>) found registered in Dbase * Team '<teamName>' (ID : <teamID>) does not have service <tagName> registered in Dbase * No service <tagName> for team '<teamName>' found in Dbase * Team <teamName> does not have adequate security level to run service <tagName> * Team '<teamName>' (ID : <teamID>) is not registered * Team '<teamName>' (ID : <teamID>) has already published service <tagName> * Team '<teamName>' (ID : <teamID>) is not registered in Dbase * No <tagName> services exist in the Dbase * An error occurred while selecting the <someNumber> <tagName> service in the Dbase * An error occurred while retrieving the number of arguments for <tagName> service (serviceID=<dbaseServiceID>) in the Dbase * • An error occurred while retrieving the number of responses for <tagName> service (serviceID=<dbaseServiceID>) in the DBase | SOA message, segments and fields okay – but their content caused some runtime issue |
| -5 | * • Error executing SQL=[<sqlStatement>] - error=[<exception thrown>] | Any exception thrown by MSSQLSERVER |

It is up to you and your team whether or not (within your Client application) you want to display error messages sent from the SOA-Registry to your user or not. You may wish to take an errorCode / errorMessage and reword it …