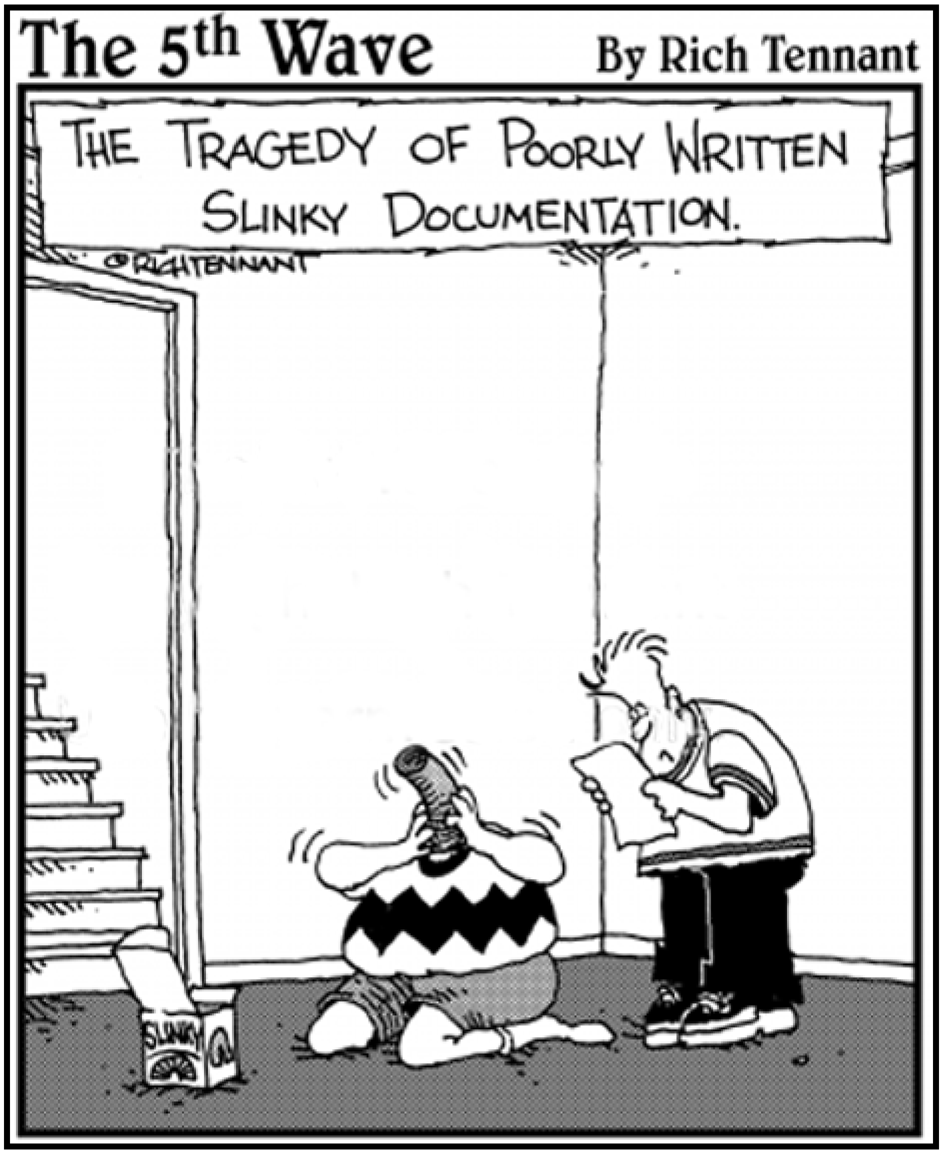
Philip GeLinas

Senior Programmer Writer

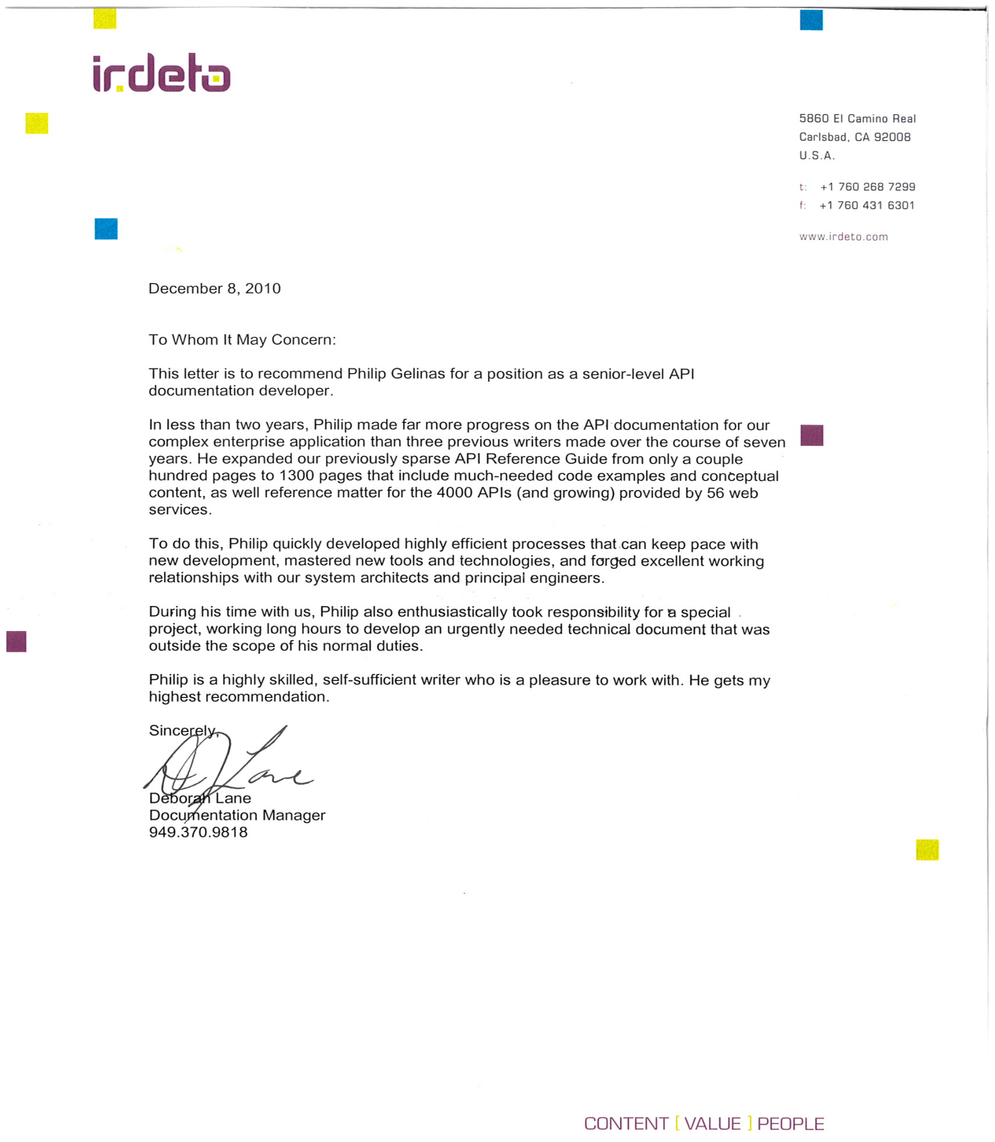
"If an API has changed
but the documentation hasn't been updated
then you can easily waste any amount of time
before you realise that the instructions you're following
don't match up to the pieces you got, IKEA-style"

—Frustrated Developer
(and former IKEA customer)


The 5th Wave
By Rich Tennant
THE TRAGEDY OF POORLY WRITTEN SLINKY DOCUMENTATION.
©RICHTENNANT
SLINKY

Reference Letter

Be sure to use the zoom feature in Acrobat to enlarge this letter enough to make it legible.



irdeto

5860 El Camino Real
Carlsbad, CA 92008
U.S.A.

t: +1 760 268 7299
f: +1 760 431 6301

www.irdeto.com

December 8, 2010

To Whom It May Concern:

This letter is to recommend Philip Gelinas for a position as a senior-level API documentation developer.

In less than two years, Philip made far more progress on the API documentation for our complex enterprise application than three previous writers made over the course of seven years. He expanded our previously sparse API Reference Guide from only a couple hundred pages to 1300 pages that include much-needed code examples and conceptual content, as well reference matter for the 4000 APIs (and growing) provided by 56 web services.

To do this, Philip quickly developed highly efficient processes that can keep pace with new development, mastered new tools and technologies, and forged excellent working relationships with our system architects and principal engineers.

During his time with us, Philip also enthusiastically took responsibility for a special project, working long hours to develop an urgently needed technical document that was outside the scope of his normal duties.

Philip is a highly skilled, self-sufficient writer who is a pleasure to work with. He gets my highest recommendation.

Sincerely,

[illegible]

Deborah Lane
Documentation Manager
949.370.9818

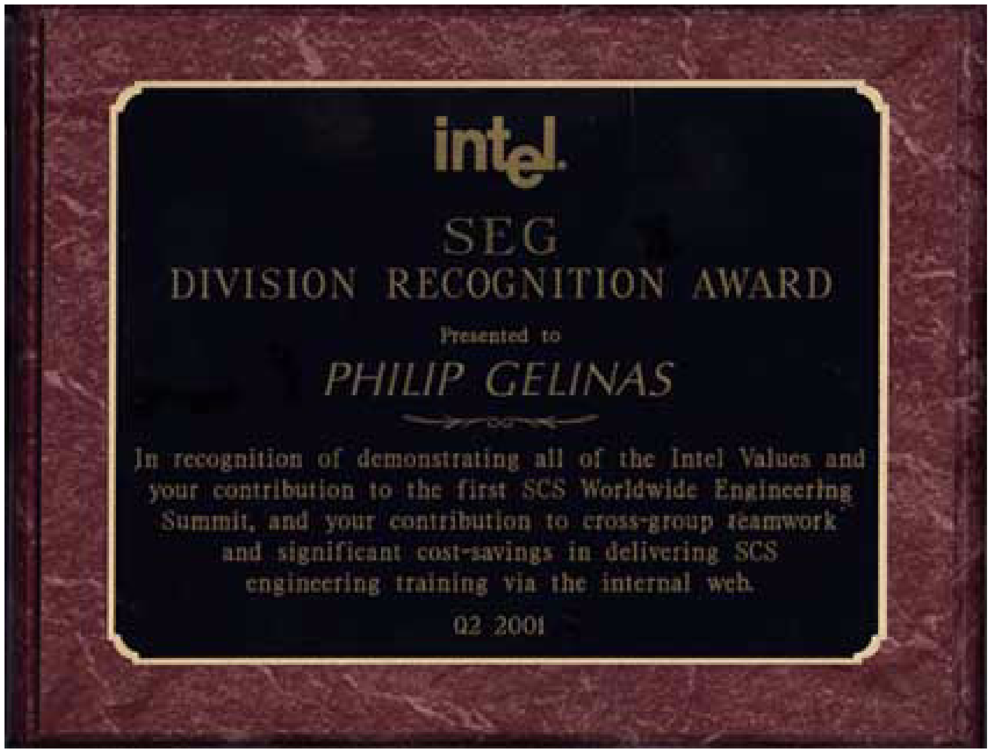
CONTENT [VALUE] PEOPLE

Awards

Intel SEG Division Recognition Award

I received this award on behalf of my team while I was program manager at Intel. We implemented a company-wide distance learning program, from conception to completion, in six weeks. Our program enabled the entire company to benefit from quarterly engineering summits that take place around the world.

One engineer from each business group across the company attends the summit and presents the latest progress, innovations, and breakthroughs on behalf of his or her group. Our online training course combined and synchronized their powerpoint presentations with videos from their lectures, which we securely streamed to over 30,000 engineers world-wide. As the program manager on this project, it was my responsiblity to assemble and lead a team of project managers, videographers, and network specialists to achieve this goal. The award was presented to us with a cash bonus, and Intel featured us in the company newsletter, which has an audience of over 70,000 employees, world-wide.



intel.

SEG
DIVISION RECOGNITION AWARD

Presented to

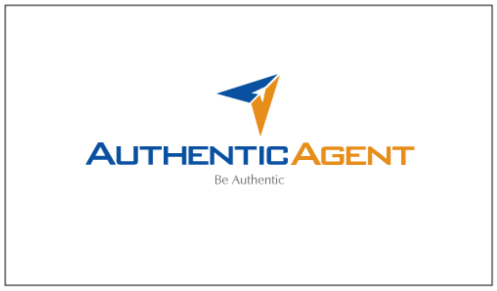
PHILIP GELINAS

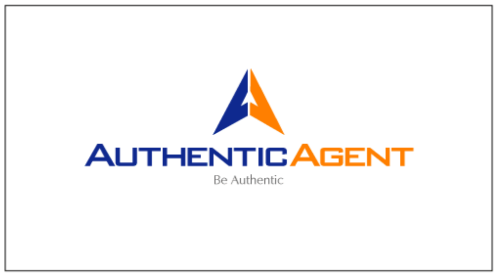
In recognition of demonstrating all of the Intel Values and your contribution to the first SCS Worldwide Engineering Summit, and your contribution to cross-group teamwork and significant cost-savings in delivering SCS engineering training via the internal web.

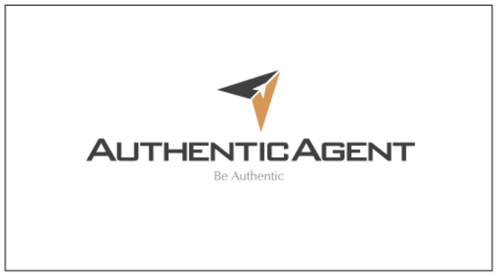
Q2 2001

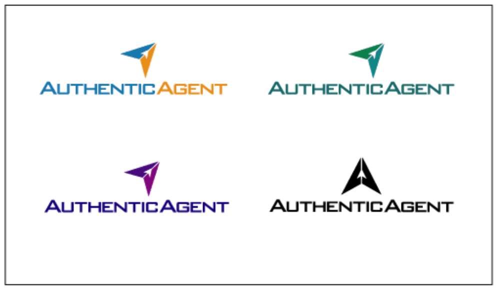
International Graphic Design Contest
1st Place, 49 Designers

I designed this logo in Adobe Illustrator and won first place in a contest with 48 other graphics artists from all around the world. Be sure to use the zoom feature in Acrobat to enlarge the image.



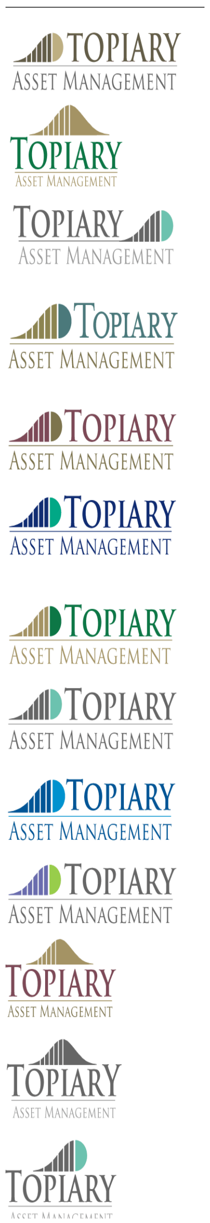






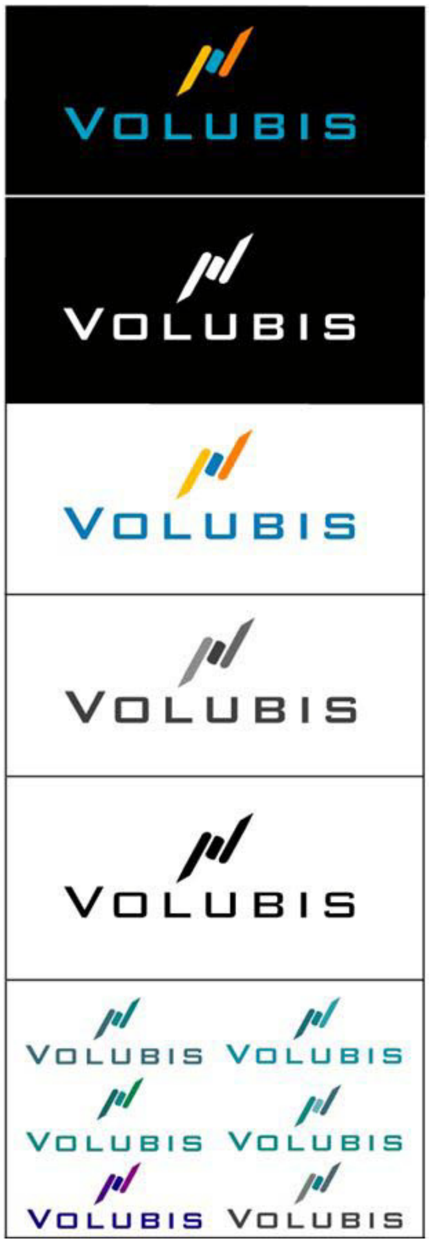
International Graphic Design Contest
1st Place, 62 Designers

I designed this logo in Adobe Illustrator and won first place in a contest with 61 other graphics artists from all around the world. Be sure to use the zoom feature in Acrobat to enlarge the image.



International Graphic Design Contest
17th Place, Honorable Mention, 109 designers

I designed this logo in Adobe Illustrator and took 17th place in a contest with 108 other graphics artists from all around the world. Be sure to use the zoom feature in Acrobat to enlarge the image.



Developer Guides

Some of the writing samples in this section are rough approximations of the actual documentation that I published. Several of my samples are publicly available, but in some cases, I had to remove illustrations, product names, company names, tables, and entire sections, so the copyright owner would grant me permission to use excerpts in my portfolio.

xPatterns Developer’s Guide

This section contains three excerpts from the xPatterns Developer’s Guide, which I wrote from scratch.

- Building a Java Client
- Deleting Documents from Your CRN
- Setting Up Authorization

Building a Java Client

This section describes how to create an xPatterns Web Services Java client.

Requirements

- Java IDE (Eclipse, Netbeans, etc...)
- xPatterns Java wrapper (<REMOVED>)
- API Key — An Atigeo representative will provide login credentials to the xPatterns Admin Portal, where you can manage your API Key.

Note You can download the xPatterns Java wrapper here: <REMOVED>.

To build a Java client

1. Start your IDE (Eclipse, Netbeans, etc...)
2. Create a Java application.
3. Once your new project is set up, download the xPatterns Java wrapper (<REMOVED>), import it to your project, and you're ready to start coding.
4. Next, in order to test the web service, we're going to call getUrlTerms on a wiki page.

```
//JAVA Example Code -- GetUrlTerms
// This example demonstrates how retrieve url terms relevant to the specific url provided.
package com.xrelevance.xpatterns_examples.examples;

import java.util.Arrays;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.web.client.xPatternsRestTemplate;

import com.xpatterns.xPatternsRestClientException;
import com.xpatterns.security.api.authorization.AuthConstants;
import com.xpatterns.xrelevance.api.model.ContentTerm;

public class GetUrlTermsExample {
        public static void main(String[] args) {

        // Optional. Set proxyHost and proxyPort to
        // view requests and responses in Fiddler.
        System.setProperty("http.proxyHost", "127.0.0.1");
        System.setProperty("http.proxyPort", "8888");

        String contentId = "AmazonBooks30";
    String url = "http://en.wikipedia.org/wiki/Seattle_Seahawks";

        String queryUrl = "<REMOVED>/xrelevance-api-rest/relevance/urlTerms/content/" +
        contentId + "?" + "url=" + url;

        // Provides a container for the xPatterns API Key.
        @SuppressWarnings("unchecked")
        HttpEntity<Object> requestEntity = getEntity();

        @SuppressWarnings("unused")
        ContentTerm contentTerm = new ContentTerm();
try {
// All API calls take place via the xPatterns REST client.
xPatternsRestTemplate client = new xPatternsRestTemplate();
client.exchange(queryUrl, HttpMethod.POST, requestEntity, null);
} catch (xPatternsRestClientException e) {
System.out.println(e.toString());
}

}

// Represents the body of the request / response.
@SuppressWarnings("rawtypes")
private static HttpEntity getEntity() {

String apiKey = "ecc18b31-fc69-4bd5-b764-96e358eb6060";
HttpHeaders requestHeaders = new HttpHeaders();
requestHeaders.put(AuthConstants.API_KEY, Arrays.asList(apiKey));

// The warnings being suppressed are a Spring artifact.
// @SuppressWarnings({ "unchecked", "rawtypes" })
HttpEntity<?> requestEntity = new HttpEntity<Object>(requestHeaders);
return requestEntity;
}
}
```

Deleting Documents from Your CRN

Once your content has been uploaded and you have a contentId, you can use deleteContentDocument() to delete documents from your CRN.

The following code example demonstrates how to permanently delete a document:

```
//JAVA Example Code — DeleteContentDocument Example
// This example demonstrates delete an existing document.

package com.xrelevance.xpatterns_examples.examples;

import java.util.Arrays;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.util.Assert;
import org.springframework.web.client.xPatternsRestTemplate;
import com.xpatterns.xPatternsRestClientException;
import com.xpatterns.security.api.authorization.AuthConstants;

public class DeleteContentDocumentExampleCode {

        public static void main(String[] args) {

        // Optional. Set proxyHost and proxyPort to
        // view requests and responses in Fiddler.
        System.setProperty("http.proxyHost", "127.0.0.1");
        System.setProperty("http.proxyPort", "8888");

        String contentId = "Butch";
        String documentId = "Document 16";

        String url = "https:/<REMOVED>/xrelevance-api-rest/cms/content/"
        + contentId + "/document/" + documentId;

        String apiKey = getApiKey();
        HttpHeaders requestHeaders = new HttpHeaders();
        requestHeaders.put(AuthConstants.API_KEY, Arrays.asList(apiKey));

        xPatternsRestTemplate client = new xPatternsRestTemplate();
        @SuppressWarnings({ "unchecked", "rawtypes" })
        HttpEntity<?> requestEntity = new HttpEntity(requestHeaders);

        try {
        client.exchange(url, HttpMethod.DELETE, requestEntity, Object.class);

        } catch (xPatternsRestClientException ex) {
        System.out.println(ex.getMessage());
        ex.printStackTrace();
        Assert.isTrue(false);

        }
        }
}
```

Setting Up Authorization

In order to make API calls, you must have an active partner account on xPatterns. You must also have a valid partner user with administrative privileges associated with your partner account. This user has the administrative rights necessary to add more users to the system.

All partner users have unique usernames and passwords, which they'll use to login to the admin portal.

API Rights

The admin portal provides a mechanism to manage access to the xPatterns public methods. There are two levels of access. One is at the partner level and the other is at the partner user level. A partner is typically an Atigeo partner. A partner user is typically someone employed by the Atigeo partner.

There are currently about 80 methods in the xPatterns public interface. Each one has two sets of API Rights associated with it: partner rights and partner user rights.

Partner Rights

Partner rights are at the partner level and control access on a method-by-method basis for all users in the system. No user can access methods that are turned off at the partner level.

Partner User Rights

Partner user rights is the other set of API rights, which control access at the partner user level. This means one partner can have multiple partner users who each have access to a different set of public methods. So, it's possible that your company has access to public methods, which are turned off for you at the partner user level. A partner user with admin access can change your privileges.

A typical partner user might be a software engineer who needs to make API calls to xPatterns. As an admin user, you'll login into the Admin portal and create a partner user account for your engineer. The system automatically generates an API key, which you'll give to the engineer to enable him or her to make API calls to xPatterns. The next section steps you through this process.

xPatterns Computer-Assisted Coding Developer’s Guide

This section contains three excerpts from the xPatterns Computer-Assisted Coding Developer’s Guide, which I wrote from scratch.

- xPatterns Computer-Assisted Coding Developer’s Guide
- Getting Acquainted with JSON
- Getting Acquainted with REST

xPatterns Computer-Assisted Coding Developer's Guide

Welcome to the xPatterns Computer-Assisted Coding (C.A.C.) Developer's Guide! This guide provides a practical introduction to developing applications for analyzing and coding of patient electronic medical records (EMRs).

The documentation provided here describes the major features of the platform and explores the concepts behind computer-aided coding. It also discusses the framework for constructing an application and the tools for developing, testing, and publishing software for the platform.

This guide is divided into several sections:

- Setting Up a Java Development Environment
  Describes how to set up your development environment so you can begin working with the REST APIs immediately.
- Getting Acquainted with REST
  The REST API is the underlying interface for all APIs in the xPatterns CAC platform. This chapter discusses some of the details of the xPatterns REST implementation.
- Getting Acquainted with JSON
  xPatterns uses JavaScript Object Notation for data transfer. This section provides a more detail description of how xPatterns uses JSON to represent data.

See Also For API specifications, see the API Reference Guide

Getting Acquainted with JSON

JSON stands for JavaScript Object Notation. It's an alternative to XML that is superior for applications that move lots of data. The main advantage of JSON is that it has a simpler structure, which makes it easier for both computers and humans to read.

This means that a JSON representation of data normally requires very little if any JSON-specific code to be processed. In fact, since JSON is built into both JavaScript and Python, no additional code is necessary at all when parsed by applications written in these languages.

This chapter describes the main features of JSON and provides some typical examples, which demonstrate how xPatterns uses JSON to transmit data.

See Also For a more detailed description of the JSON standard, visit http://www.json.org/.

JSON Structures

JSON is built upon two universal data structures.: A collection of key-value pairs and ordered lists of values.

Note The character encoding of JSON text is always Unicode.

Key-Value Pairs

JSON is built around the idea of key-value pairs. A key-value pair is a set of two pieces of data that are linked together.

A key is the unique identifier for the piece of data, and the value is either the data itself or a pointer to the location of that data.

A value can be a string in double quotes, a number, true or false, null, an object, or an array. These structures can be nested.

Ordered Lists of Values

An array is an ordered collection of values. An array begins with square brackets. Commas separate multiple values within the brackets.

JSON Response

Here's an example response from xPatterns to a getContent() request. You can see that the general structure of the JSON in this response is based around key/value pairs and lists of things. We're going to refer to this example several times in the following sections as we discuss the basic data types that JSON supports.

```
{
"keywords":[
        "test1",
        "test2",
        "test3"
],
"dynamicFields":
{
        "d.long":20,
                    "a.string":"asdf",
                    "f.double":2.66,
                    "e.float":2.4,
                    "c.boolean":true,
                    "b.integer":3},

        "body":"test add document through queue",
        "id":"1",
        "title":"my document",
        "source":"653dcb46-433a-409da989-c4f4a0eb4421",
        "description":"document content"}
```

Getting Acquainted with REST

This chapter provides a quick overview of the REST protocol, which explains how to make HTTP requests to the xPatterns platform.

- Overview of REST.<Link Removed>
- Common Characteristics of xPatterns REST Resources
- Types of xPatterns API requests. <Link Removed>
- How the API Reference Guide is organized. <Link Removed>

Common Characteristics of xPatterns REST Resources

The following characteristics apply to all xPatterns API resources:

- You access a resource by sending an HTTP request to the xPatterns API server. The server replies with a response that either contains the data you requested, or a status indicator, or both.
- You request a particular resource by appending a particular path to this base URL that specifies the resource.
- All resources may return any of the status codes listed in HTTP Status Codes. Other status or odes that a particular resource might return are listed specifically in the section of this page that describes that resource.
- On this page, when a portion of a URL, path, or parameter value is shown in italics, this indicates that you should replace the italicized value with a particular value appropriate to your request.

<REMOVED> Developer's Guide

Windows Communication Foundation

The Windows Communication Foundation (WCF) is Microsoft's unified framework for building secure, reliable, transacted, and interoperable distributed applications. The WCF (.NET 3.0) programming model unifies Web Services, .NET Remoting, Distributed Transactions, and Message Queues into a single Service-oriented programming model for distributed computing. <REMOVED> uses WCF to establish service endpoints for communication with the Web services.

This chapter discusses the following topics:

- The WCF Service Model Endpoints
- Namespaces
- Interfaces
- Lookup Lists
- ServiceFactory
- AuthenticationHeader
- GetServiceLocation

See Also For more information about Windows Communication Foundation, visit Microsoft's developer website: http://msdn.microsoft.com/en-us/netframework/aa663324.aspx

The WCF Service Model Endpoints

<REMOVED> uses the following service endpoints for communication with the Web services.

- **Address**—where the service is located. Addess will always point to a .svc file hosted by IIS.
- **Binding**—how the client communicates. This value is always "basicHttpBinding".
- **Contract**—what the service can do. Contract will always point to the interface in the assembly which exposes the public routines for accessing the business objects.

  <endpoint

  name="I<***entity name***>Service"
  address=http://<REMOVED>interprit/ASM/ALL/<***entity name***>.svc
  binding="basicHttpBinding"
  contract="PayMedia.ApplicationServices.<***entity name***>.ServiceContracts.
  I<***entity name***>Service"

  />

Namespaces

Most assemblies have the following logical structure:

- **PayMedia.ApplicationServices.[Entity]. ServiceContracts**—this namespace contains the Interfaces which expose business and configuration data (including LookupLists and Enumerations).
- **PayMedia.ApplicationServices.[Entity].ServiceContracts.DataContracts**—this namespace defines the data structures and contains the major business objects.

Note In a typical <REMOVED> installation, assemblies are located here:
C:\Inetpub\wwwroot\ASM\ALL\bin

Interfaces

In general, each ServiceContract namespace groups two Interfaces:

- I[Entity]Service—provides methods for interacting with business data. For example, ICustomersService provides methods such as CreateCustomer(), GetCustomer(), and UpdateCustomer().
- I[Entity]ConfigurationService—provides methods for interacting with configuration data. For example, ICustomersConfigurationService provides methods such as CreateValidAddress(), GetKeywordTypes(), and GetProvinceByCountry().

Lookup Lists

Most business entities in <REMOVED> require lists of configuration data that are stored in Lookup Lists (e.g., Title, Customer Type, Country, Currency); conceptually, Lookup Lists are dynamically-configured enumerations.

All <REMOVED> configuration services provide a GetLookups(LookupLists lookupList)) method that will retrieve whatever collections of data is in those lists.

The GetLookups() method takes one LookupList value as a parameter, which identifies the list, and returns a collection of values from the corresponding Lookup List in the Configuration module.

The LookupCollections contain Lookup objects, which always have a Key and Description property that identify and describe each piece of data.

For example, using the CustomersConfigurationService, invoking:

- **GetLookups(LookupLists.Title)** returns the collection of valid customer titles (e.g., Mr., Mrs., Miss, Company, etc) that is stored in configuration.
- **GetLookups(LookupLists.Country)** returns the collection of customer countries that is stored in configuration.

```
public static LookupCollection lkpAddrTitleColl;
//Gets Lookup List of Customer Titles
if (lkpAddrTitleColl == null)
{
lkpAddrTitleColl = CustConfigService.GetLookups(
PayMedia.ApplicationServices.Customers.ServiceContracts.LookupLists.Title
);
```

ServiceFactory

PayMedia.ApplicationServices.Factory returns WCF service proxies. We can either store the URI and authentication data in an app.config file, or we can send it through one of the overloaded versions of GetService().

GetService<> is a generic method that takes an Interface type as a parameter and returns that Interface as a working proxy. The GetService() method is overloaded to support several configuration options.

AuthenticationHeader

Each <REMOVED> Web service requires user credentials, and each Method authenticates a user.

When calling the GetService() method, we send an AuthenticationHeader containing a username, password, and dsn.

The ServiceFactory creates instances of service proxies and preserves the credentials within each service. The first call to GetService will be to create an instance of the AuthenticationService service proxy. An AuthenticationHeader object is passed to the ServiceFactory, but Authentication has not yet taken place. If we pass in invalid credentials, we will not be notified at this point.

AuthenticateByProof() does a formal authentication, and if authentication is successful, an AuthenticationResult type is returned. This object contains an Authenticated (boolean) property which indicates success or failure. This object also contains an encrypted Token property that we can use as a substitute for the Proof in our AuthenticationHeader object.

By eliminating the password from the AuthenticationHeader, we provide greater security across method calls. Every method call authenticates the user and resets the time stamp for the Token.

GetServiceLocation

Using configuration files to store the URIs of the ASM layer prevents the need to recompile and redeploy. However, if the ASM layer is moved, many configuration changes will be necessary, so we recommend that you store the URIs in a database that your application can query as it loads.

```
string uri = IServiceLocationService.GetServiceLocation(string interfaceName);
```

The GetServiceLocation() method of the IServiceLocationService will retrieve the fully qualified URIs of the web services when given the Interface name of the service. In Oracle, this data is located in the Systemdata schema, within the ServiceLocation table. Once the URI has been retrieved, the Service Proxy is created by the Service Factory as before.

Example Code

The following example illustrates how to configure an Authentication procedure.

1. From C:\Inetpub\wwwroot\ASM\ALL\bin, add references to the assemblies that you want to access. For example,
   - PayMedia.ApplicationServices.Authentication.ServiceContracts.dll
   - PayMedia.ApplicationServices.Customers.ServiceContracts.dll
   - PayMedia.ApplicationServices.Finance.ServiceContracts.dll
   - PayMedia.ApplicationServices.InvoiceRun.ServiceContracts.dll
   - PayMedia.ApplicationServices.SharedContracts.dll
   - PayMedia.ApplicationServices.Factory.dll
2. Open the app.config file and set WCF endpoints for Authentication. For example:
   - Customers
   - CustomersConfiguration
   - Finance
   - FinanceConfiguration
3. Create a new (e.g., frmLogon.cs), and add the Authentication procedure. This procedure should create an AuthenticationHeader object and use that object to create a local IAuthenticationService interface.
4. Assign textbox values from the Logon Form to the UserName, Proof, and Dsn properties of the AuthenticationHeader.
5. Raise an Exception if the user fails to logon successfully.
6. Create the GetServices routine.
7. Use the ServiceFactory to create local Service Interfaces. For example:
   - ICustomersService, ICustomersConfigurationService,
   - IFinanceService, and IFinanceConfigurationService.

```
using System;
using System.Collections;
using System.Windows.Forms;
using PayMedia.ApplicationServices.Customers.ServiceContracts;
using PayMedia.ApplicationServices.Finance.ServiceContracts;
using PayMedia.ApplicationServices.InvoiceRun.ServiceContracts;
using PayMedia.ApplicationServices.Authentication.ServiceContracts;
using PayMedia.ApplicationServices.SharedContracts;
using PayMedia.ApplicationServices.Factory;
using PayMedia.ApplicationServices.Authentication.ServiceContracts.DataContracts;
using PayMedia.ApplicationServices.Customers.ServiceContracts.DataContracts;

namespace EnvironAndConfig
{
   public partial class frmLogon : Form
{
   # region variables
```

```
ICustomersService CustService;
ICustomersConfigurationService CustConfigService;
IFinanceService FinService;
IFinanceConfigurationService FinConfigService;
IInvoiceRunConfigurationService InvoiceRunConfigService;
IAuthenticationService AuthService;
AuthenticationHeader authHeader;

#endregion

public frmLogon()
{
  InitializeComponent();
}

private void btnLogon_Click(object sender, EventArgs e)
{
  if (Authenticate())
  {
    GetServices();
    txtUserName.Enabled = false;
    txtPassword.Enabled = false;
    txtDSN.Enabled = false;
    btnLogon.Enabled = false;
  }
}

//Creates the AuthenticationHeader for logging in
private bool Authenticate()
{
  try
  {
            //TODO: Create an AuthenticationHeader. Assign values from the controls to the\
            //UserName, Proof, and Dsn properties.
    authHeader = new AuthenticationHeader();
    authHeader.UserName = txtUserName.Text;
    authHeader.Proof = txtPassword.Text;
    authHeader.Dsn = txtDSN.Text;
                //TODO: Create a local IAuthenticationService Interface. Pass the
                //AuthenticationHeader to the ServiceFactory.
    AuthService = ServiceFactory.GetService<IAuthenticationService>(authHeader);
    DsnCollection dsns = new DsnCollection();
    dsns.Add(new Dsn(txtDSN.Text));
    UserIdentity identity = new UserIdentity(txtUserName.Text, dsns);
            //TODO: Call the AuthenticateByProof() method of the IAuthenticationService
            //Interface and pass in the identity and password of the current user. The return
            //value should be stored in an AuthenticationResult object.
    AuthenticationResult result = AuthService.AuthenticateByProof(identity, txtPassword.Text, string.Empty);
            //TODO: Check the value of the AuthenticationResult's Authenticated
            //property to see if the user was authenticated. If not, raise an exception. If
            //authenticated, display a confirmation.
    if (result.Authenticated == false)
    {
      throw new Exception(string.Format("Authentication Failed for User {0}", txtUserName.Text));
    }
    else
    {
            //As an additional security measure, you can replace the password in the
            //AuthenticationHeader with the returned token
      authHeader.Proof = "";
      authHeader.Token = result.Token;
     MessageBox.Show("Logged In as User: " + result.UserName, "Logon Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
                grpFindCustomer.Enabled = true;
                return true;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            return false;
        }
    }
    //Retrieves whatever services we need
    private void GetServices()
    {
        //TODO: Use the ServiceFactory to create local service interfaces for ICustomersService,
        //ICustomersConfigurationService, IFinanceService, IFinanceConfigurationService, and
        //IInvoiceRunConfigurationService
        CustService = ServiceFactory.GetService<ICustomersService>(authHeader);
        CustConfigService = ServiceFactory.GetService<ICustomersConfigurationService>(authHeader);
        FinService = ServiceFactory.GetService<IFinanceService>(authHeader);
        FinConfigService = ServiceFactory.GetService<IFinanceConfigurationService>(authHeader);
        InvoiceRunConfigService = ServiceFactory.GetService<IInvoiceRunConfigurationService>(authHeader);
    }

    private void btnGetCustomer_Click(object sender, EventArgs e)
    {
                    //TODO: Create a Customer object and test the Customer Service by passing a valid
                    //Customer ID to the GetCustomer() method
        Customer cust = CustService.GetCustomer(int.Parse(txtCustomerID.Text));
        MessageBox.Show("Customer's first name is: " + cust.DefaultAddress.FirstName, "Customer Data", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
  }
}
```

This section discusses the following topics:

- Retrieve an AddressCollection
- Identify Address Types
- Retrieve ParkingRules from Configuration
- Identify ParkingRule values
- Retrieve a FinancialAccountCollection.

Addresses and AddressCollections

Address information is fundamental to the Customer capture process, and the CRUD functions we perform here will be done similarly on other entities. Addresses are stored in an AddressCollection.

When retrieving Address information, use the GetAddresses() method of the ICustomersService Interface, passing in the CustomerID

```
//Gets all Addresses for a Customer
if (addColl == null)
addColl = CustService.GetAddresses(int.Parse(txtCustomerID.Text), null,
0);
```

✓ **Note** GetAddresses() takes a CriteriaCollection as a parameter, and this type can be used to filter the returned results of the operation. For filtering to work, the types being returned must be marked up and compiled with Criteria attributes. At the time of this writing, AddressType is the only property that can be used as a criterion for Address entities. Ex., new Criteria(Key="Type", Value="908") will return just the Shipping address (908 is the enum value of "Shipping").

Address Types

The AddressTypes are an Enumeration, not configurable, due to <REMOVED> identifying these as the most commonly used in the industry.

The ICustomersService.GetAddressByType() method will return an Address object based on the AddressType passed to the method. This method also requires an EntityID passed, and if no entity with that ID value can be found, the Default address is returned. Although "Work Order" is listed as a "peer" level AddressType, it is only used when creating a work order. It is not generally seen in the initial stages of the Customer capture process.

- Each customer has one Default Address and seven other types of additional Addresses
- The AddressCollection can be searched for individual Addresses based on the AddressType Enumeration.

Supplied Logic

To focus specifically on the API, several routines have been pre-supplied to handle much of the implementation logic. CreateLookupTable() is located in frmLogon. It takes a LookupCollection and a (string) TableName parameter and stores the Key and Description properties into a DataTable. Doing so allows the more "readable" Descriptions to be presented on the controls while making the required Key values easily accessible for CRUD functions.

GetAllLookups() and GetReasonLookups() are used to bind LookupList data to controls on the forms. SetParkingRules() is provided as a part of the given logic because of its complexity. SetParkingRules() accepts three parameters: 1) A RuleCollection, which is retrieved by called the "GetRules()" method on any service and passing in the name of the method which will enforce those rules, (in our application, this is done as part of the GetCustomerData() method on frmLogon); 2) A Control.ControlCollection, which allows any TabPage's control collection to be verified against the parking rules, and 3) A String, which takes the "Property" property prefix of the entity being checked. (look at source code and ASMWSTester). SetParkingRules is called by lstEnableAddressOperation and lstEnableAcctOperation (the two small listboxes on the form that the user clicks in order to enable add/update/delete operations).

Parking Rules

Parking Rules are defined "per operation". Each operation has rules about what data can, cannot, and must be modified. Those rules are set in Configuration, and they are enforced whenever that operation is called. For example, when updating a Customer's address, we call the "UpdateAddress()" method on the ICustomersService interface. If "BigCity" is required, then we must supply a value for the BigCity entry when Updating an Address. In order to find out what the parking rules are for any given method, we have to pass the name of that method as a parameter of the GetRules() method for the service which supports that operation. In the case of "UpdateAddress", which is supported by the ICustomersService interface, the call would look like this:

RuleCollection rc = ICustomersService.GetRules("UpdateAddress");

GetRules() returns a RuleCollection, and each Rule contains a Parking property of type ParkingRule (enum). The ParkingRule enumeration lists four possible values: Editable, ReadOnly, Required, and SystemGenerated (a database-generated value).

Retrieving Parking Rules

RuleCollections are retrieved by calling the GetRules() method of the Service which hosts the operation and passing in the name of that operation.

```
public static RuleCollection custParkingRulesColl;
custParkingRulesColl = CustService.GetRules("CreateAddress");
foreach (PayMedia.ApplicationServices.SharedContracts.Rule pkngRule in
frmLogon.custParkingRulesColl){
switch (pkngRule.Parking){
case ParkingRule.Editable:
break;
case ParkingRule.ReadOnly:
break;
case ParkingRule.Required:
break;
case ParkingRule.SystemGenerated:
break;
}
}
```

Here is an example of the overall structure for retrieving the Parking Rules of the CreateAddress method in the ICustomersService interface. These rules are checked/ enforced when UpdateAddress is called. A similar procedure would be used to find the parking rules for other methods. Since the Required value is useful for enforcing validation rules, it may be useful to store the required controls in a data structure. Required = In the Configuration module when a field is set to "Must Enter" Editable = In the Configuration module when a field is set to "May Park"

ReadOnly = In the Configuration module when a field is NOT set to "May Park"

SystemGenerated = When there is no setting in the Configuration module (e.g., address.Id) The GetRules() method is defined on the (higher-level) IASMCommon Interface

Retrieving Financial Accounts

Retrieving Financial Accounts is similar to retrieving Addresses. Retrieving the FinancialAccountCollection is a fairly straightforward process: call GetFinancialAccountsForCustomer() and pass in the CustomerID

- Service: IFinanceService
- Method: GetFinancialAccountsForCustomer()
- Returns: FinancialAccountCollection
- Details: Include Properties, Configuration LookupLists, and predefined Enumerations.

```
FinAccColl = FinService.
GetFinancialAccountsForCustomer(int.Parse(txtCustomerID.Text), null, 0);
```

Unlike Addresses, Financial Accounts can not be retrieved by Type. They can only be retrieved based on the FinancialAccountId (using GetFinancialAccountsByIds()) or the CustomerId (using GetFinancialAccountsForCustomer()) to which they belong.

lstEnableAcctOperation works like the list box in the Address demo. It enables/disables the buttons, sets the parking rules for Financial Accounts, and gets the Lookup List reason data per operation. The "Get Accounts" button populates the "List of Accounts" combo box with the list of Financial Account ID's.

The "Add" and "Update" buttons work like those in the Address form. They validate the control data against the parking rules and the Error Provider control displays where required data is omitted. The SelectedIndexChanged() event procedure of the ListOfAccounts combo box displays the account details of the selected FinancialAccountID. The SelectedIndexChanged() event procedure of the MOP combo box displays the parking rules associated strictly with the Method Of Payment

Retrieving Customer Data

This section discusses the following topics:

- Retrieve an AddressCollection
- Identify Address Types
- Retrieve ParkingRules from Configuration
- Identify ParkingRule values
- Retrieve a FinancialAccountCollection.

Addresses and AddressCollections

Address information is fundamental to the Customer capture process, and the CRUD functions we perform here will be done similarly on other entities. Addresses are stored in an AddressCollection.

When retrieving Address information, use the GetAddresses() method of the ICustomersService Interface, passing in the CustomerID

```
//Gets all Addresses for a Customer
if (addColl == null)
addColl = CustService.GetAddresses(int.Parse(txtCustomerID.Text), null,
0);
```

✓ **Note** GetAddresses() takes a CriteriaCollection as a parameter, and this type can be used to filter the returned results of the operation. For filtering to work, the types being returned must be marked up and compiled with Criteria attributes. At the time of this writing, AddressType is the only property that can be used as a criterion for Address entities. Ex., new Criteria(Key="Type", Value="908") will return just the Shipping address (908 is the enum value of "Shipping").

Address Types

The AddressTypes are an Enumeration, not configurable, due to <REMOVED> identifying these as the most commonly used in the industry.

The ICustomersService.GetAddressByType() method will return an Address object based on the AddressType passed to the method. This method also requires an EntityID passed, and if no entity with that ID value can be found, the Default address is returned. Although "Work Order" is listed as a "peer" level AddressType, it is only used when creating a work order. It is not generally seen in the initial stages of the Customer capture process.

- Each customer has one Default Address and seven other types of additional Addresses
- The AddressCollection can be searched for individual Addresses based on the AddressType Enumeration.

Supplied Logic

To focus specifically on the API, several routines have been pre-supplied to handle much of the implementation logic. CreateLookupTable() is located in frmLogon. It takes a LookupCollection and a (string) TableName parameter and stores the Key and Description properties into a DataTable. Doing so allows the more "readable" Descriptions to be presented on the controls while making the required Key values easily accessible for CRUD functions.

GetAllLookups() and GetReasonLookups() are used to bind LookupList data to controls on the forms. SetParkingRules() is provided as a part of the given logic because of its complexity. SetParkingRules() accepts three parameters: 1) A RuleCollection, which is retrieved by called the "GetRules()" method on any service and passing in the name of the method which will enforce those rules, (in our application, this is done as part of the GetCustomerData() method on frmLogon); 2) A Control.ControlCollection, which allows any TabPage's control collection to be verified against the parking rules, and 3) A String, which takes the "Property" property prefix of the entity being checked. (look at source code and ASMWSTester). SetParkingRules is called by lstEnableAddressOperation and lstEnableAcctOperation (the two small listboxes on the form that the user clicks in order to enable add/update/delete operations).

Parking Rules

Parking Rules are defined "per operation". Each operation has rules about what data can, cannot, and must be modified. Those rules are set in Configuration, and they are enforced whenever that operation is called. For example, when updating a Customer's address, we call the "UpdateAddress()" method on the ICustomersService interface. If "BigCity" is required, then we must supply a value for the BigCity entry when Updating an Address. In order to find out what the parking rules are for any given method, we have to pass the name of that method as a parameter of the GetRules() method for the service which supports that operation. In the case of "UpdateAddress", which is supported by the ICustomersService interface, the call would look like this:

RuleCollection rc = ICustomersService.GetRules("UpdateAddress");

GetRules() returns a RuleCollection, and each Rule contains a Parking property of type ParkingRule (enum). The ParkingRule enumeration lists four possible values: Editable, ReadOnly, Required, and SystemGenerated (a database-generated value).

Retrieving Parking Rules

RuleCollections are retrieved by calling the GetRules() method of the Service which hosts the operation and passing in the name of that operation.

```
public static RuleCollection custParkingRulesColl;
custParkingRulesColl = CustService.GetRules("CreateAddress");
foreach (PayMedia.ApplicationServices.SharedContracts.Rule pkngRule in
frmLogon.custParkingRulesColl){
switch (pkngRule.Parking){
case ParkingRule.Editable:
break;
case ParkingRule.ReadOnly:
```

```
break;
case ParkingRule.Required:
break;
case ParkingRule.SystemGenerated:
break;
}
}
```

Here is an example of the overall structure for retrieving the Parking Rules of the CreateAddress method in the ICustomersService interface. These rules are checked/ enforced when UpdateAddress is called. A similar procedure would be used to find the parking rules for other methods. Since the Required value is useful for enforcing validation rules, it may be useful to store the required controls in a data structure. Required = In the Configuration module when a field is set to "Must Enter" Editable = In the Configuration module when a field is set to "May Park"

ReadOnly = In the Configuration module when a field is NOT set to "May Park"

SystemGenerated = When there is no setting in the Configuration module (e.g., address.Id) The GetRules() method is defined on the (higher-level) IASMCommon Interface

Retrieving Financial Accounts

Retrieving Financial Accounts is similar to retrieving Addresses. Retrieving the FinancialAccountCollection is a fairly straightforward process: call GetFinancialAccountsForCustomer() and pass in the CustomerID

- Service: IFinanceService
- Method: GetFinancialAccountsForCustomer()
- Returns: FinancialAccountCollection
- Details: Include Properties, Configuration LookupLists, and predefined Enumerations.

```
FinAccColl = FinService.
GetFinancialAccountsForCustomer(int.Parse(txtCustomerID.Text), null, 0);
```

Unlike Addresses, Financial Accounts can not be retrieved by Type. They can only be retrieved based on the FinancialAccountId (using GetFinancialAccountsByIds()) or the CustomerId (using GetFinancialAccountsForCustomer()) to which they belong.

lstEnableAcctOperation works like the list box in the Address demo. It enables/disables the buttons, sets the parking rules for Financial Accounts, and gets the Lookup List reason data per operation. The "Get Accounts" button populates the "List of Accounts" combo box with the list of Financial Account ID's.

The "Add" and "Update" buttons work like those in the Address form. They validate the control data against the parking rules and the Error Provider control displays where required data is omitted. The SelectedIndexChanged() event procedure of the ListOfAccounts combo box displays the account details of the selected FinancialAccountID. The SelectedIndexChanged() event procedure of the MOP combo box displays the parking rules associated strictly with the Method Of Payment

<Product Name REMOVED>Developer's Guide

Processing XML Documents

This chapter contains code examples that demonstrate how to process XML documents. It provides guidelines for extracting values from XML elements and attributes contained in XML output. This chapter includes the following sections:

XML Processing

One typical function that your calling application must perform is to search for a particular node (element or attribute) within an XML document and extract its value. Some XML nodes have the same name, but appear in two or more locations in the XML document (under different parents). Because new nodes will be added to the data model schema as Fair Isaac enhances Decision Accelerator functionality, it is important to search by using full path names as opposed to just the node name.

This chapter contains a Java code snippet for a small application that processes an XML document. The sample document (TestDocument.xml) is an excerpt of a typical Decision Accelerator output XML document, containing root, CreditRequest, Applicant, and DecisionResponse elements.

✓ **Note** If you have questions about XML processing, contact Fair Isaac Product Support.

Test XML Document

The following XML document, TestDocument.xml, was used to test the example code shown in Figure 2 on page 37. The document contains multiple ApplicantTelephone, UserDefinedField, and ReasonText elements.

```
<Application DeliveryOptionCode="sysid" ApplicationType="Individual"
DecisioningRequestType="Final Decision" ProcessingRequestType="DA"
TransactionId="-1062731253:86c347:100bfc5cf3b:-8000" Timestamp="2005-10-17T09:30:47-
05:00">
   <MessageList StatusCode="0" StatusDescription="Successful"/>
   <DecisionResponse>
      <Product>
         <Decision DecisionResult="Decline">
            <Reason ReasonCode="AB">
               <ReasonText>gross total debt service ratio - insufficient information
               available to calculate</ReasonText>
            </Reason>
            <Reason ReasonCode="CD">
               <ReasonText>time at present address too short or unknown</ReasonText>
            </Reason>
            <Reason ReasonCode="EF">
               <ReasonText>present employment status</ReasonText>
            </Reason>
         </Decision>
      </Product>
   </DecisionResponse>
   <CreditRequest ProductCategory="HomeEquity" LoanPurpose="Renovation"
   ProductCode="HELOCRevolving"/>
   <Applicant ApplicantCrossReferenceId="222444888" ApplicantType="Primary">
      <Personal PersonalIdNumber="444558888" Birthdate="1957-11-15"/>
      <ApplicantTelephone PhoneNumber="4138887777" PhoneType="Home"/>
      <ApplicantTelephone PhoneNumber="4138187171" PhoneType="Mobile"/>
      <UserDefinedField>
         <Name>Customer Value Indicator</Name>
         <DataType>string</DataType>
         <Value>Gold</Value>
      </UserDefinedField>
      <UserDefinedField>
         <Name>In-house Deposit Balance</Name>
         <DataType>real</DataType>
         <Value>785.50</Value>
      </UserDefinedField>
   </Applicant>
</Application>
```

Figure 1: The TestDocument.xml file

Processing Code

The example code shown in Figure 2 demonstrates how to process an XML document using the *Xalan-Java* XML processor. Xalan-java builds on the *DOM* (Document Object Model) level 2 specification. The code evaluates XPath expressions to extract values. To build an XPath expression for a particular XML node, you must know the location of the XML node in the Decision Accelerator data model.

This code uses Xalan-Java version 2.6.0. The file xalan-j-current-bin-2jars.zip can be downloaded from http://xml.apache.org/xalan-j/downloads.html. Out of the JARs included in this download, there are three which you must include in the *classpath*:

- xalan.jar
- xsltc.jar
- xercesImpl.jar

See Also

- For more information about Xalan-Java, see http://xml.apache.org/xalan-j/index.html.
- For more information about the DOM, see http://www.w3.org/DOM/.
- For more information about XPath, see http://www.w3.org/TR/xpath.
- For more information about the Decision Accelerator data model, see Chapter 3, “The Capstone Origination Suite Data Model.”

In some XML documents, an element may be repeated; each instance of the element may have different attribute values. The test document contains multiple ApplicantTelephone elements. The code sample demonstrates how to filter on the value of the PhoneType attribute. The test document also contains some UserDefinedField elements. The processing code sample demonstrates how to filter these elements on known values of their Name attributes.

```
package Example;

import java.io.*;

import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.*;
import org.xml.sax.*;
import org.apache.xml.serialize.XMLSerializer;
import org.apache.xpath.*;

public class ParseXml
{
   /* main method receives an array of String */
   public static void main(String[] args)
   {
      if (args.length == 1)
```

```
        (new ParseXml()).parseFile(args[0]);
    else
        System.out.println("Pass the name of the file to parse.");
}

Document _doc;

/* creates a new instance of ParseXml */
public ParseXml()
{
}

private
void
/* the name fo the file must be an absolute path */
parseFile(String fileName)
{
    try
    {
        /* parse xml to dom */
        DOMParser
        parser = new DOMParser();

        parser.parse(new InputSource(new FileReader(fileName)));

        _doc = parser.getDocument();

        showOutput();
    }
    catch(Exception Ex)
    {
        System.out.println(Ex.getMessage());
    }
}

/* all XPath expressions are in reference to the XML document (_doc) */
private void showOutput() throws Exception
{
    System.out.println("Credit Request Information:");
    System.out.println("    Decisioning Request Type: " + applyXPath("/Application/
    @DecisioningRequestType"));
    System.out.println("    Product Category: " + applyXPath("/Application/
    CreditRequest/@ProductCategory"));
    System.out.println("    Loan Purpose: " + applyXPath("/Application/
    CreditRequest/@LoanPurpose"));
    System.out.println("    Transaction ID: " + applyXPath("/Application/
    @TransactionId"));
    System.out.println("    Timestamp: " + applyXPath("/Application/@Timestamp"));

    System.out.println();
    System.out.println("Applicant Information:");
    System.out.println("    Applicant ID: " + applyXPath("/Application/Applicant/
    @ApplicantCrossReferenceId"));
    System.out.println("    Personal ID Number: " + applyXPath("/Application/
    Applicant/Personal/@PersonalIdNumber"));
    System.out.println("    Birthdate: " + applyXPath("/Application/Applicant/
    Personal/@Birthdate"));

    System.out.println();
    System.out.println("Telephone Information:");
    System.out.println("    Home: " + applyXPath("/Application/Applicant/
    ApplicantTelephone[@PhoneType='Home']/@PhoneNumber"));
    System.out.println("    Mobile: " + applyXPath("/Application/Applicant/
        ApplicantTelephone[@PhoneType='Mobile']/@PhoneNumber"));

        System.out.println();
        System.out.println("Customer Information:");
        System.out.println("    Customer Value Indicator: " + applyXPath("/Application/
        Applicant/UserDefinedField[Name = 'Customer Value Indicator']/Value/text()"));
        System.out.println("    In-house Deposit Balance: " + applyXPath("/Application/
        Applicant/UserDefinedField[Name = 'In-house Deposit Balance']/Value/text()"));

        System.out.println();
        System.out.println("Decision Result: " + applyXPath("/Application/
        DecisionResponse/Product/Decision/@DecisionResult"));
        System.out.println();
        System.out.println("Decision Reasons: ");
        String [] reasons = applyXPathMulti("/Application/DecisionResponse/Product/
        Decision/Reason/ReasonText/text()");
        for(int i = 0, iEnd = reasons.length; i < iEnd; ++i)
        System.out.println("    " + reasons[i]);
    }

    /* returns a single node */
    private String applyXPath(String xpath) throws Exception
    {
        Node
        node = XPathAPI.selectSingleNode (_doc, xpath);

        return  (node != null)
                 ?   node.getNodeValue()
                 :   "";
    }
    /* returns a node list, which is copied into an array of String */
    private String[] applyXPathMulti(String xpath) throws Exception
    {
        NodeList
        nodes = XPathAPI.selectNodeList(_doc, xpath);

        String[] results = new String[nodes.getLength()];

        for(int i = 0, iEnd = results.length; i < iEnd; ++i)
            results[i] = nodes.item(i).getNodeValue();

        return results;
    }

}
```

Figure 2: Java code for processing an XML document <REMOVED>

Processing Output

Figure 3 shows the result of processing the TestDocument.xml file using the code shown in Figure 2.

```
Credit Request Information:
    Decisioning Request Type: Final Decision
    Product Category: Home Equity
    Loan Purpose: Renovation
    Transaction ID: -1062731253:86c347:100bfc5cf3b:-8000
    Timestamp: 2005-10-17T09:30:47-05:00

Applicant Information:
    Applicant ID: 222444888
    Personal ID Number: 444558888
    Birthdate: 1957-11-15

Telephone Information:
    Home: 4138887777
    Mobile: 4138187171

Customer Information:
    Customer Value Indicator: Gold
    In-house Deposit Balance: 785.50

Decision Result: Decline

Decision Reasons:
    gross total debt service ratio - insufficient information available to calculate
    time at present address too short or unknown
    present employment status
```

Figure 3: XML processing results <REMOVED>

<Product Name REMOVED> Developer's Guide

Transaction Data and Decisioning

Before calling systems can submit transactions to Decision Accelerator to be decisioned, SMW users must configure decision flows to account for the types of requests that will be submitted. SMW users must provide calling system developers with information about the input and output data used by the decision flows. This chapter provides information on various Decision Accelerator decision flow components and their relation to the IBS data model and transaction data. It includes the following sections:

- Context Paths
- Data Object References
- Data Methods
- Data Methods and Data Object References Used in Decisioning
- Cross-Reference IDs
- Data Method History Output
- Processing History Output
- Using User-Defined Fields
- Resubmitting Transactions
- Internal Services
- Web Services
- Updating the Data Repository with Final Decision and Offer Data
- Using Decision Accelerator with Intelligent Data Manager

Context Paths

The data model schema represents the elements and attributes that can be used in XML input and output. The same schema, when imported into Blaze Advisor, represents the data objects and fields that can be used for decisioning and the relationships among these items.

In XPath and when referring to XML documents, parent and child elements are separated by slashes, and an attribute is separated from its containing element by "/@". For example:

- Application/Finance/Income/Owner
- Application/Finance/Income/Owner/@OwnershipPercent

In the SMW, elements are separated from each other and from attributes by dots (periods). The SMW equivalents of the paths above are:

- Application.Finance.Income.Owner
- Application.Finance.Income.Owner.OwnershipPercent

Decision flow components such as data methods, data object references, rules, and mappings can only access data within the *context* specified for the component. To specify a context in the SMW, you must choose the desired object from the Data Object Browser.

Data Object References

Data submitted for decisioning (instantiated objects) must conform to the data model schema. This actual data can have multiple occurrences of some objects (for example, multiple ApplicantAddress objects), while the schema only indicates that multiple ApplicantAddress objects are allowed. A data object reference can be used to specify which particular instances of objects should be used for decisioning. A data object reference returns a list of objects that satisfy a set of conditions; it can return multiple objects if multiple instances match the specified conditions.

For example, if you create a data object reference to look at previous addresses for the applicant, and the application input contains multiple previous addresses, the data object reference returns multiple previous addresses. If the data object reference is used as the context of a rule, when the rule executes, it operates on every object instance referred to by the data object reference.

The Data Object Reference Editor in the SMW features a Data Object Browser, which displays the imported data model in a tree structure. The root element (Application) is the root node of the tree, and each child element is shown indented one level from its parent. When a data object reference is added, it is shown as a leaf of its associated object.

Data Methods

Data methods are functions that can be authored in the SMW. When executed, they perform computational tasks, such as arithmetic operations, string manipulations, date/time field manipulations, and object creation and handling. They can be used to generate the data needed in business policies (rules). For example, data methods can be used to compute and return values in transaction output, to aggregate income for an individual, to compute the age of a piece of data given a timestamp, or to find the maximum value of a set of values. They can be used to attach data objects to each other, detach data objects, and call internal services or Web services.

The *Data Method Editor* in the SMW also uses the *Data Object Browser*. When used for authoring data methods, the Data Object Browser displays data object references and data methods that have already been created.

For testing purposes, Decision Accelerator can return information on the data methods executed during the execution of a decision flow. For more information on data method output, see "Data Method History Output" on page 44.

✓ **Note** Decision Accelerator automatically creates only the objects listed in section 8 of "Decision Accelerator Data Model Constraints" on page 314. If you want other objects returned in transaction output (for example Offer and Booking objects under Product), you must create and attach those objects using data methods.

Data Methods and Data Object References Used in Decisioning

The following example shows how data object references and data methods can be used together to aggregate data submitted in multiple objects into a single value for decisioning.

In the StandardCard product in the pre-defined product strategies delivered with Decision Accelerator, some of the rules are based on applicant income. The LowMonthlyIncome rule compares the primary applicant's monthly income to a constant. The rule uses the data object reference dor_PrimaryApplicant and the data method dm_App_WeightedMonthlyIncome.

The data method dm_App_WeightedMonthlyIncome calculates the weighted sum of monthly income for an applicant, given stated ownership percentage. Because the data model allows income items to be entered with various frequencies (such as weekly, monthly, quarterly, and yearly), the data method calls a second data method (dm_Util_MonthlyAmount) that converts the amount from the given frequency to the amount for monthly frequency. The for loop of the data method uses the context path (Application.Finance.Income.Owner) defined by the dor_AllIncomeOwner data object reference.

Figure 4 on page 42 shows an example of some input Income nodes for an application with two applicants. Each income item is cross-referenced to either the primary or co-applicant. To use database terminology, the Owner/@ApplicantCrossReferenceId attribute acts as a "foreign key", referring back to the "primary key" (the ApplicantCrossReferenceId attribute of the Applicant object).

```
<Application>
   …
   <Applicant ApplicantCrossReferenceId="1" ApplicantType="Primary"/>
   <Applicant ApplicantCrossReferenceId="2" ApplicantType="CoApplicant"/>
   <Finance>
      <Income IncomeType="Primary" IncomeAmount="7200" Frequency="Annually"
      SourceType="Self-employed" VerifiedFlag="false" StartDate="1985-05-07"
      IgnoreIncomeItemFlag="false">
         <Owner ApplicantCrossReferenceId="1"/>
      </Income>
      <Income IncomeType="Primary" IncomeAmount="330" Frequency="Bi-weekly"
      SourceType="Primary job" VerifiedFlag="true" VerificationDate="2004-08-15"
      StartDate="2001-03-26" IgnoreIncomeItemFlag="false">
         <Owner ApplicantCrossReferenceId="2"/>
      </Income>
      <Income IncomeType="Ancillary" IncomeAmount="300" Frequency="Quarterly"
      SourceType="Investment" VerifiedFlag="false" IgnoreIncomeItemFlag="false">
         <Owner ApplicantCrossReferenceId="1"/>
      </Income>
      <Income IncomeType="Ancillary" IncomeAmount="40" Frequency="Weekly"
      SourceType="Non job-related" VerifiedFlag="false" StartDate="2002-11-02"
      AnticipatedEndDate="2006-01-01" IgnoreIncomeItemFlag="true">
         <Owner ApplicantCrossReferenceId="1" OwnershipPercent="75"/>
         <Owner ApplicantCrossReferenceId="2" OwnershipPercent="25"/>
      </Income>
   </Finance>
   …
</Application>
```

Figure 4: Income nodes for two applicants

The dm_Util_MonthlyAmount data method converts the income amounts for each applicant as follows:

Applicant	Income amount	Frequency	Conversion factor	Monthly amount
1	$7200.00	Yearly	amount / 12	$600.00
1	$300.00	Quarterly	amount / 3	$100.00
1	$40.00	Weekly	amount * 4.33	$173.20
2	$330.00	Bi-weekly	amount * 2.17	$716.10
2	$40.00	Weekly	amount * 4.33	$173.20

The dm_App_WeightedMonthlyIncome data method calculates the weighted monthly income for each applicant as follows:

Applicant	Monthly amount	Ownership %	Weighted amount
1	$600.00	n/a	$600.00
1	$100.00	n/a	$100.00
1	$173.20	75	$129.90
Total			**$829.90**
2	$716.10	n/a	$716.10
2	$173.20	25	$43.30
Total			**$759.40**

The context of the LowMonthlyIncome rule uses the dor_PrimaryApplicant data object reference. The rule checks to see whether the weighted monthly income of the primary applicant ($829.90) is less than a constant value ($500). In this case, the *condition* of the rule is not met and the rule does not fire.

Cross-Reference IDs

This example in the previous section demonstrates the importance of using cross-reference IDs consistently throughout transaction input; without them, the pre-defined product strategies delivered with Decision Accelerator cannot identify the correct owners of each income item, and they cannot accurately perform aggregate calculations.

The value of Application/Applicant/@ApplicantCrossReferenceId should be an exact string match to the following attributes:

- Application/CreditRequest/@ProceedsDeliveryApplicantCrossReferenceId
- The ApplicantCrossReferenceId attribute of:
  - Application/Business/Owner
  - Application/ExternalDataSource/BbDataSource/BbDataServiceRequest/BbRequest/BbApplicant
  - Application/ExternalDataSource/<REMOVED>DataSource/<REMOVED>DataServiceRequest/<REMOVED>Request/Requested<REMOVED>/<REMOVED>InputArf/ApplicantCrossReference
  - Application/ExternalDataSource/<REMOVED>DataSource/<REMOVED>DataServiceRequest/<REMOVED>Request/<REMOVED>Applicant
  - Application/ExternalDataSource/<REMOVED>DataSource/<REMOVED>RequestDefaults/DefaultRequested<REMOVED>/<REMOVED>InputArf/ApplicantCrossReference
  - Application/ExternalDataSource/FesDataSource/FesDataServiceRequest/FesRequest/FesApplicant
  - Application/ExternalDataSource/FesDataSource/FesRequestDefaults/DefaultFesApplicant
  - Application/ExternalDataSource/SbfeDataSource/SbfeDataServiceRequest/SbfeRequest/SbfeApplicant
  - Application/Finance/Asset/Owner
  - Application/Finance/Income/Owner
  - Application/Finance/Liability/Owner

Similarly, the value of Application/Business/@BusinessCrossReferenceId should be an exact string match to the BusinessCrossReferenceId attribute of the following elements:

- Application/ExternalDataSource/BbDataSource/BbDataServiceRequest/LosRequest/LosBusiness
- Application/ExternalDataSource/BbDataSource/BbDataServiceRequest/BbRequest/RequestedBb/BbInputArf
- Application/ExternalDataSource/BbDataSource/BbDataServiceRequest/BbRequest/BbBusiness
- Application/ExternalDataSource/BbDataSource/BbRequestDefaults/DefaultRequestedBb/BbInputArf
- Application/ExternalDataSource/SbfeDataSource/SbfeDataServiceRequest/SbfeRequest/SbfeBusiness
- Application/ExternalDataSource/SbfeDataSource/SbfeDataServiceRequest/SbfeRequest/SbfeApplicant/SbfeBusinessApplicant

Important Values must match exactly, and are case-sensitive. The introduction of as little as an extra leading or trailing space can cause problems when an application is decisioned.

Data Method History Output

For testing and debugging purposes, Decision Accelerator can return detailed information on data methods executed during decisioning. This information appears in DataMethodHistory output elements. All objects in the IBS data model used for Decision Accelerator contain DataMethodHistory children, with the following exceptions:

- DataMethodHistory objects and their descendants
- UserDefinedField objects and their descendants

As described in "Complex and Simple Elements" on page 64, simple elements (which may contain content such as report images, ARF text, or reason text) do not correspond to objects in the data model. Therefore, they do not contain DataMethodHistory children.

DataMethodHistory elements are not displayed in the Outline or DataModel tabs of the data model spreadsheet. Instead, they appear in the DataMethodHistory tab. The paths for DataMethodHistory and its descendants are shown below.

Paths:

- `<ParentElement>/DataMethodHistory`
- `<ParentElement>/DataMethodHistory/@Timestamp`
- `<ParentElement>/DataMethodHistory/Name`
- `<ParentElement>/DataMethodHistory/DataType`
- `<ParentElement>/DataMethodHistory/Value`
- `<ParentElement>/DataMethodHistory/DataMethodInput`
- `<ParentElement>/DataMethodHistory/DataMethodInput/Name`
- `<ParentElement>/DataMethodHistory/DataMethodInput/DataType`
- `<ParentElement>/DataMethodHistory/DataMethodInput/Value`
- `<ParentElement>/DataMethodHistory/DataMethodOutput`
- `<ParentElement>/DataMethodHistory/DataMethodOutput/Name`
- `<ParentElement>/DataMethodHistory/DataMethodOutput/DataType`
- `<ParentElement>/DataMethodHistory/DataMethodOutput/Value`

<ParentElement> is a placeholder for any element that can contain a DataMethodHistory object. An example of a DataMethodHistory element is shown in Figure 5.

```
<Application>
   <Applicant>
      <DataMethodHistory Timestamp="2005-11-03T08:22:06-08:00">
         <Name>dmo_App_GetPreviousApplicantAddress_ByIndex</Name>
         <DataType>ApplicantAddress</DataType>
         <Value>&lt;ApplicantAddress AddressStatusIndicator="Previous"
         AddressType="Home" UnparsedStreetAddress="1500 Calveros Blvd #13"
         City="Sheffield" State="MA" PostalCode="01257" PostalPlusCode="5588"
         County="" CountryCode="USA" ResidenceType="Condominium"
         ResidentialStatusIndicator="Rents" MonthsAtAddress="50"
```

```
            StartDate="1999-12-01" EndDate="2004-02-14" VerifiedFlag="true"
            VerificationType="Phone bill" VerificationDate="2005-08-09"&gt;&lt;/
            ApplicantAddress&gt;</Value>
            <DataMethodInput>
               <Name>xIndex</Name>
               <DataType>numeric</DataType>
               <Value>1</Value>
            </DataMethodInput>
            <DataMethodOutput>
               <Name>xCount</Name>
               <DataType>numeric</DataType>
               <Value>2</Value>
            </DataMethodOutput>
         </DataMethodHistory>
      </Applicant>
   </Application>
```

Figure 5: A sample DataMethodHistory output node

Note The value of the Value element matches the data type listed in the DataType element. However, you should be aware that the actual values in the Name, DataType, and Value elements are strings.

Using Data Method History Output During Testing

When creating a data method using the Data Method Editor of the Enterprise Origination Solutions, a Decision Accelerator user can select the Save History check box. When this option is selected and the data method is executed, information appears in the following output elements:

- DataMethodHistory: The output XML document contains one DataMethodHistory element for each data method that is executed during a decision flow. The DataMethodHistory element appears as a child of the data method's context object. It has the following descendants:
  - Timestamp attribute—The execution start date/time of the data method execution.
  - Name element—The data method's name.
  - DataType element—The data method's *result type*.
  - Value element—The value of the data method's "result" when the *data method expression* execution is complete. If the data method's result is an object, the Value contains an "escaped XML string" that contains an XML depiction of the result object and all its children. If the Value is null, no Value attribute is returned. A Value string of zero length appears as <Value/>.
  - One DataMethodInput element for each data method input variable, and one DataMethodOutput element for each data method *local variable*.
- DataMethodHistory/DataMethodInput: For each DataMethodHistory element in the XML output, there are 0-n DataMethodInput elements, one for each of the *input* parameters (variables) passed to the data method when it was executed. The DataMethodInput element has the following children:
  - Name element—The input variable's name.
  - DataType element—The input variable's data type.

  - Value element—The value of the input variable, as passed to the data method (that is, before the data method's expression was executed). If the input is an object, the Value contains an "escaped XML string" that contains an XML depiction of the input object and all its children. If the Value is null, no Value attribute is returned. A Value string of zero length appears as <Value/>.
- DataMethodHistory/DataMethodOutput: For each DataMethodHistory element in the XML output, there are 0-n DataMethodOutput elements, one for each of this data method's local variables.
  - Name element—The local variable's name.
  - DataType element—The local variable's data type.
  - Value element—The value of the local after the data method expression has been executed. If the local is an object, the Value contains an "escaped XML string" that contains an XML depiction of the local object and all its children. If the Value is null, no Value attribute is returned. A Value string of zero length appears as <Value/>.

Data Method History Examples

Each of the following four examples illustrates how a data method would appear in the Data Method Editor and how it could be called. Each example contains a snippet of the input XML document sent for decisioning, and a snippet of the output document that shows the resulting *data method history*.

1. Simple Method, No Input Variables/No Local Variables

Table 1: Data Method Editor Input

Item	Value
Data Method Name	dm_Liability_IsBankruptcy
Description	Returns a true if the Liability is a bankruptcy
Context (Path+Element)	Application.Finance.Liability
Result Type	boolean
Save History	checked
Input Variables	none
Local Variables	none

Expression	result := false;
	if (DerogatoryType = "Bankruptcy-Type7") then result := true; endif
	if (DerogatoryType = "Bankruptcy-Type11") then result := true; endif
	if (DerogatoryType = "Bankruptcy-Type12") then result := true; endif
	if (DerogatoryType = "Bankruptcy-Type13") then result := true; endif

Method Calling

This method can be called from rules, mappings, or other methods and returns a boolean value. Here is an example of calling this method from another method:

Table 2: Data Method Call

Item	Value
Expression	SomeField := dm_Liability_IsBankruptcy

Sample XML Input Document

```
<Application>
   …
   <Finance>
      <Liability LiabilityTypeCode="MRTG" LiabilityAmount="364000"
      MonthlyPaymentAmount="2150" OurAccountFlag="true">
         <Owner ApplicantCrossReferenceId="1" />
         <Owner ApplicantCrossReferenceId="2" />
      </Liability>
      <Liability DerogatoryType="Bankruptcy-Type12" DerogatoryEventDate="1999-08-23"
      OurAccountFlag="false">
         <Owner ApplicantCrossReferenceId="1" />
      </Liability>
   </Finance>
   …
</Application>
```

Figure 6: The liability objects sent for evaluation

Saved Data Method History Output

The result is equal to “true” because one of the liabilities is a bankruptcy.

```
<Application>
   …
   <Finance>
      <Liability LiabilityTypeCode="MRTG" LiabilityAmount="364000"
      MonthlyPaymentAmount="2150" OurAccountFlag="true">
         <Owner ApplicantCrossReferenceId="1" />
         <Owner ApplicantCrossReferenceId="2" />
         <DataMethodHistory Timestamp="2005-10-14T17:55:06.253-08:00">
            <Name>dm_Liability_IsBankruptcy</Name>
            <DataType>boolean</DataType>
            <Value>false</Value>
         </DataMethodHistory>
      </Liability>
      <Liability DerogatoryType="Bankruptcy-Type12" DerogatoryEventDate="1999-08-23"
      OurAccountFlag="false">
         <Owner ApplicantCrossReferenceId="1" />
         <DataMethodHistory Timestamp="2005-10-14T17:55:06.253-08:00">
            <Name>dm_Liability_IsBankruptcy</Name>
            <DataType>boolean</DataType>
            <Value>true</Value>
         </DataMethodHistory>
      </Liability>
   </Finance>
   …
</Application>
```

Figure 7: The output created when the data method is called at runtime

2. Simple Method, One Input Variable/No Local Variables

Table 3: Data Method Editor Input

Item	Value		
Data Method Name	dm_Root_ConcatenateVipNumber		
Description	Sets the Application VipNumber		
Context (Path+Element)	Application		
Result Type	string		
Save History	checked		
Input Variables	**Name**	**Data Type**	**Initial Value**
	xString	string	n/a
Local Variables	none		

Expression	VipNumber := VipNumber + xString;
	result := VipNumber		

Method Calling

This method can be called from other methods, but not from *data method sequences*, rules, or mappings. It returns a string value. Here is an example of calling this method from another method.

Table 4: Data Method Call

Item	Value
Expression	dm_Root_ConcatenateVipNumber ("34");

Sample XML Input Document

```
<Application SubmitterId="MySubmitterId" DeliveryOptionCode="sysid"
ApplicationType="Joint" DecisioningRequestType="Final Decision"
ProcessingRequestType="DA" ResubmissionFlag="false" PromotionCode="LF0808"
VipNumber="12"/>
```

Figure 8: The document containing the initial VipNumber

Saved Data Method History Output

The data method concatenates “34” to the initial value of “12” to create a result of “1234”.

```
<Application SubmitterId="MySubmitterId" DeliveryOptionCode="sysid"
ApplicationType="Joint" DecisioningRequestType="Final Decision"
ProcessingRequestType="DA" ResubmissionFlag="false" PromotionCode="LF0808"
VipNumber="1234"/>
   <DataMethodHistory Timestamp="2005-10-15T08:22:06.253-08:00">
      <Name>dm_Root_ConcatenateVipNumber</Name>
      <DataType>string</DataType>
      <Value>1234</Value>
      <DataMethodInput>
         <Name>xString</Name>
         <DataType>string</DataType>
         <Value>34</Value>
      </DataMethodInput>
   </DataMethodHistory>
</Application>
```

Figure 9: The output created when the data method is called at runtime

3. Complex Method, No Input Variables/No Local Variables

Table 5: Data Method Editor Input

Item	Value
Data Method Name	dmo_App_GetCurrentAddress
Description	Get current address for this applicant.
Context (Path+Element)	Application.Applicant
Result Type	ApplicantAddress
Save History	checked
Input Variables	none
Local Variables	xAppCurAddr

Expression	result := null;
	
	for every Application.Applicant.ApplicantAddress relative to Applicant do
	    if (AddressStatusIndicator = "Current") then
	        xAppCurAddr := ApplicantAddress;
	    endif
	endfor
	
	result := xAppCurAddr;

Method Calling

This method can be called from rules or other methods, but not from mappings. It returns a copy of the current applicant address. Here is an example of calling this method from another method.

Table 6: Data Method Call

Item	Value		
Local Variables	**Name**	**Data Type**	**Initial Value**
	xAppAddress	ApplicantAddress	null

Expression	xAppAddress := dmo_App_GetCurrentAddress;
	SomeField := xAppAddress.City;		

Sample XML Input Document

```
<Application>
   <Applicant>
      <ApplicantAddress AddressStatusIndicator="Current" AddressType="Home"
      StreetNumber="256" StreetName="Morton" StreetType="Dr"
      StreetPostDirection="NW" City="Sandisfield" State="MA" PostalCode="01255"
      PostalPlusCode="0413" County="Berkshire" CountryCode="USA"
      ResidenceType="Single Family Home" ResidentialStatusIndicator="Owns
      (Financed)" MonthsAtAddress="26" StartDate="2004-02-15" EndDate=""
      VerifiedFlag="false"/>
      <ApplicantAddress AddressStatusIndicator="Previous" AddressType="Home"
      UnparsedStreetAddress="1500 Calveros Blvd #13" City="Sheffield" State="MA"
      PostalCode="01257" PostalPlusCode="5588" County="" CountryCode="USA"
      ResidenceType="Condominium" ResidentialStatusIndicator="Rents"
      MonthsAtAddress="50" StartDate="1999-12-01" EndDate="2004-02-14"
      VerifiedFlag="true" VerificationType="Phone bill" VerificationDate="2005-08-
      09"/>
      <ApplicantAddress AddressStatusIndicator="Previous" AddressType="Home"
      UnparsedStreetAddress="7855 Atwood Ave. #6" City="Sheffield" State="MA"
      PostalCode="01256" PostalPlusCode="" County="" CountryCode="USA"
      ResidenceType="Apartment" ResidentialStatusIndicator="Rents"
      MonthsAtAddress="10" StartDate="1999-01-15" EndDate="1999-11-30"
      VerifiedFlag="true" VerificationType="Phone bill" VerificationDate="2005-08-
      09"/>
   </Applicant>
</Application>
```

Figure 10: An input document containing a current address and previous addresses

Saved Data Method History Output

Only the current address is returned as the result of executing the data method.

```
<Application>
   <Applicant>
      <DataMethodHistory Timestamp="2005-11-02T08:22:06.253-08:00">
         <Name>dmo_App_GetCurrentApplicantAddress</Name>
         <DataType>ApplicantAddress</DataType>
         <Value>&lt;ApplicantAddress AddressStatusIndicator="Current"
         AddressType="Home" StreetNumber="256" StreetName="Morton" StreetType="Dr"
         StreetPostDirection="NW" City="Sandisfield" State="MA" PostalCode="01255"
         PostalPlusCode="0413" County="Berkshire" CountryCode="USA"
         ResidenceType="Single Family Home" ResidentialStatusIndicator="Owns
         (Financed)" MonthsAtAddress="26" StartDate="2004-02-15" EndDate=""
         VerifiedFlag="false"&gt; &lt;/ApplicantAddress&gt;</Value>
      </DataMethodHistory>
   </Applicant>
</Application>
```

Figure 11: The output created when the data method is called at runtime

4. Complex Method, One Input Variable/One Local Variable

Table 7: Data Method Editor Input

Item	**Value**		
Data Method Name	dmo_App_GetPreviousApplicantAddress_ByIndex		
Description	Returns a copy of applicant’s Nth previous address		
Context (Path+Element)	Application.Applicant		
Type	ApplicantAddress		
Save History	checked		
Input Variables	**Name**	**DataType**	**Initial Value**
	xIndex	numeric	n/a

Table 7: Data Method Editor Input (continued)

Item	Value		
Local Variables	**Name**	**DataType**	**Initial Value**
	xCount	numeric	0

Expression	result := null;
	for every Application.Applicant.ApplicantAddress relative to Applicant do
	  if (AddressStatusIndicator = "Previous") then
	    xCount := xCount + 1;
	    if (xCount = xIndex) then
	      result := ApplicantAddress;
	    endif
	  endif
	endfor		

Method Calling

This method can be called from other methods, but not from data method sequences, mappings, or rules. It returns a copy of the ApplicantAddress. Here is an example of calling this method from another method.

Table 8: Data Method Call

Item	Value		
Local Variables	**Name**	**Data Type**	**Initial Value**
	`xAppAddress`	`ApplicantAddress`	`null`

Expression	xAppAddress := dmo_App_GetPreviousApplicantAddress_ByIndex(1);
	SomeField := xAppAddress.City;		

Sample XML Input Document

The sample input for this example is the same as for the previous example (Figure 10 on page 50).

Saved Data Method History Output

The Index value of 1 was set in the method call. On execution, two previous addresses were counted.

```
<Application>
   <Applicant>
      <DataMethodHistory Timestamp="2005-11-03T08:22:06.253-08:00">
         <Name>dmo_App_GetPreviousApplicantAddress_ByIndex</Name>
         <DataType>ApplicantAddress</DataType>
         <Value>&lt;ApplicantAddress AddressStatusIndicator="Previous"
         AddressType="Home" UnparsedStreetAddress="1500 Calveros Blvd #13"
         City="Sheffield" State="MA" PostalCode="01257" PostalPlusCode="5588"
         County="" CountryCode="USA" ResidenceType="Condominium"
         ResidentialStatusIndicator="Rents" MonthsAtAddress="50" StartDate="1999-12-
         01" EndDate="2004-02-14" VerifiedFlag="true" VerificationType="Phone bill"
         VerificationDate="2005-08-09"&gt;&lt;/ApplicantAddress&gt;</Value>
         <DataMethodInput>
            <Name>xIndex</Name>
            <DataType>numeric</DataType>
            <Value>1</Value>
         </DataMethodInput>
```

```
            <DataMethodOutput>
               <Name>xCount</Name>
               <DataType>numeric</DataType>
               <Value>2</Value>
            </DataMethodOutput>
         </DataMethodHistory>
      </Applicant>
   </Application>
```

Figure 12: The output created when the data method is called at runtime

Processing History Output

The ProcessingHistory output node contains information on the product strategy and decision flow components that were executed during transaction processing. You can use this information to help develop and maintain product strategies in your development and production environments. Table 9 describes the elements that appear in this node.

Table 9: ProcessingHistory Elements

Element	Purpose
DataMethodSequenceHistory	Exists for each data method sequence that was executed during decision flow or decision tree processing. Indicates the data method sequence name.
DecisionFlowHistory	Exists for each decision flow that was executed during transaction processing. Its descendants contain information on the decision flow components (steps) executed during processing of a decision flow.
DecisionFlowStepHistory	Exists for each decision flow component (step) executed during processing of a decision flow. Indicates the step's name, index, implementation type, implementation name, result, start timestamp, and stop timestamp.
DecisionHistory	Exists for each decision registered during transaction processing. Indicates the decision result (level). Decisions can be registered during processing of rulesets, decision scenarios, score models, and decision trees.
DecisionReason	Exists for each decision reason associated with a ruleset decision, decision scenario, score model, or decision tree.
DecisionScenarioHistory	Exists for each decision scenario that was executed during execution of a decision tree. Indicates the decision scenario name.
DecisionTableHistory	Exists for each decision table that was executed during decision flow or decision tree processing. Indicates the table name and result.
DecisionTreeHistory	Exists for each decision tree that was executed during decision flow or decision tree processing. Indicates the decision tree result.
FinalApplicationDecision	(Reserved for future use) Contains information on the final application decision of the calling system (as opposed to the application decision recommended by <REMOVED>). This object is not generated by <REMOVED> at runtime.

Table 9: ProcessingHistory Elements (continued)

Element	Purpose
FinalDecisionHistory	Exists if a a final decision was reached for the application. Indicates the final decision result.
FinalOfferHistory	(Reserved for future use) This object is not generated by <REMOVED> at runtime.
OfferHistory	Exists for each offer created during execution of a pricing scenario. Indicates the name, rate, credit limit, and term of the offer.
PricingScenarioHistory	Exists for each pricing scenario that was executed during execution of a decision tree. Its descendants contain offer and final offer information.
RuleHistory	Exists for each rule that fired during decision flow processing. Indicates the rule name, reason associated with the rule, and rank order of the reason.
RulesetHistory	Exists for each ruleset executed during decision flow processing. Indicates the ruleset name, ruleset result, and total severity of rules fired in the ruleset.
ScoreModelCharHistory	Exists for each scored characteristic evaluated in a score model. Indicates the name, bin, and partial score of the characteristic.
ScoreModelHistory	Exists for each score model executed during decision flow or scoring scenario processing. Indicates the score model name, final score, and odds.
ScoreModelRowHistory	Exists for each score model information row in a scoring scenario. Indicates the row index, its cutoff range, and whether the row was the first one executed in a scenario. Its descendants contains score model, characteristic, and candidate decision information.
ScoringScenarioHistory	Exists for each scoring scenario that was executed during execution of a decision tree. Indicates the scoring scenario name and decision result (level). Its descendants contain information on score model information rows.

See Also Refer to the data model spreadsheet for detailed information on the elements and attributes in the ProcessingHistory node.

Using User-Defined Fields

You can use UserDefinedField objects for data that doesn't have a home in the data model provided with DecisionAccelerator.

User-defined fields can be used for:

- Data submitted by the calling system on input and returned on output without being accessed by Decision Accelerator
- Data submitted by the calling system on input and accessed by Decision Accelerator within a decision flow

- Data created and/or used by Decision Accelerator within a decision flow; this data can be returned to the calling system if desired

All objects in the IBS data model used for Decision Accelerator contain UserDefinedField children, with the following exceptions:

- DataMethodHistory objects and their descendants
- UserDefinedField objects and their descendants

As described in "Complex and Simple Elements" on page 64, simple elements (which contain content such as report images, ARF text, or reason text) do not correspond to objects in the data model. Therefore, they do not contain UserDefinedField children. No schema modifications are needed for adding, deleting, or modifying UserDefinedField children of objects in the provided data model; however, if you customize the data model with new objects, you must add UserDefinedField children to the new objects if you wish to set up user-defined fields on those objects.

UserDefinedField elements are not displayed in the Outline or DataModel tabs of the data model spreadsheet. Instead, they appear in the UserDefinedField tab. The paths for UserDefinedField and its children are shown below.

Paths:

- <ParentElement>/UserDefinedField
- <ParentElement>/UserDefinedField/Name
- <ParentElement>/UserDefinedField/DataType
- <ParentElement>/UserDefinedField/Value

<ParentElement> is a placeholder for any element that can contain a UserDefinedField object. The value of the Value element must match the data type listed in the DataType element. The data types allowed for user-defined fields are:

- boolean
- date
- duration
- integer
- real
- string
- time
- timestamp

Note The actual values in the Name, DataType, and Value elements are strings.

User-Defined Fields Submitted by the Calling System

On input, user-defined fields allow you to associate a name with a value at runtime. The Name, DataType, and Value children of the UserDefinedField element must be populated. Some examples of UserDefinedField elements are shown in Figure 13.

```
<Application>
   …
   <Applicant>
      <UserDefinedField
         <Name>Customer Value Indicator</Name>
         <DataType>string</DataType>
         <Value>Gold</Value>
      </UserDefinedField>
```

```
        <UserDefinedField
            <Name>In-house Deposit Balance</Name>
            <DataType>real</DataType>
            <Value>785.50</Value>
        </UserDefinedField>
    </Applicant>
    <Business>
        <UserDefinedField
            <Name>Admin <REMOVED>t Percentage</Name>
            <DataType>integer</DataType>
            <Value>5</Value>
        </UserDefinedField>
        <UserDefinedField
            <Name>Non-Profit Indicator</Name>
            <DataType>boolean</DataType>
            <Value>true</Value>
        </UserDefinedField>
    </Business>
    <UserDefinedField
        <Name>Promotion Name</Name>
        <DataType>string</DataType>
        <Value>Tent Sale</Value>
    </UserDefinedField>
    <UserDefinedField
        <Name>Promotion End Date</Name>
        <DataType>date</DataType>
        <Value>2005-02-12</Value>
    </UserDefinedField>
</Application>
```

Figure 13: Sample UserDefinedField nodes at various levels

Documenting Your User-Defined Fields

You are responsible for maintaining documentation of any user-defined XML structures used in your transactions. For example, if your transaction contains some codes in UserDefinedField elements, you would need to maintain a list of the code meanings.

```
<UserDefinedField
    <Name>Code1</Name>
    <DataType>string</DataType>
    <Value>A52</Value>
</UserDefinedField>
<UserDefinedField
    <Name>Code2</Name>
    <DataType>string</DataType>
    <Value>B97</Value>
</UserDefinedField>
```

Figure 14: An example of user-defined fields that need to be documented

Table 10 shows an example of how the user-defined field values in Figure 14 could be documented.

Table 10: Sample User-Defined Field Documentation

Code	Meaning
A52	Fraud alert
B97	Security attention

As another example, if your transactions contain user-defined fields with duration values, you might need to document the units of the duration, such as months or years.

```
<UserDefinedField
   <Name>TimeSinceDefault</Name>
   <DataType>real</DataType>
   <Value>5.5</Value>
</UserDefinedField>
<UserDefinedField
   <Name>TimeSinceLastPayment</Name>
   <DataType>integer</DataType>
   <Value>8</Value>
</UserDefinedField>
```

Figure 15: User-defined fields with duration values<REMOVED>

Table 11 shows an example of how the user-defined field values in Figure 15 could be documented.

Table 11: Sample User-Defined Duration Field Documentation

Element	Data Type	Units	Range
`TimeSinceDefault`	`real`	`Years`	`0 - 99`
`TimeSinceLastPayment`	`integer`	`Months`	`0 - 120`

Alternately, you could imply units in the Name attributes that you assign, as shown in Figure 16.

```
<UserDefinedField
   <Name>YearsSinceDefault</Name>
   <DataType>real</DataType>
   <Value>5.5</Value>
</UserDefinedField>
<UserDefinedField
   <Name>MonthsSinceLastPayment</Name>
   <DataType>integer</DataType>
   <Value>8</Value>
</UserDefinedField>
```

Figure 16: User-defined fields with names that imply durations

User-Defined Field Data Accessed by Decision Accelerator

In order for Decision Accelerator to access a user-defined field, there must be a pre-existing agreement between the calling system owners and the Decision Accelerator owners as to the user-defined field location, name, data type, and possible values.

The actual data type of the Value element is always a string, even if the value of the DataType element is numeric, timestamp, or duration. In this case, you must use a data method to convert the value from string to the desired type.

This does not apply to data object references, since they simply return the object. If the data needs to be type converted, a data method is the best choice for selecting and converting the value.

Using User-Defined Fields in Data Methods

Most of the Decision Accelerator decision flow components where you will use user-defined field values (for example, rules and mappings) cannot call data methods that take parameters, so you will need to create a unique data method for each user-defined field Name and DataType pair, for each parent data type (object type).

Figures 17 through 20 show examples of various types of user-defined fields; tables 12 through 15 show examples of data methods that use them.

```
<Application>
   <Applicant>
      <UserDefinedField>
         <Name>ABC</Name>
         <DataType>string</DataType>
         <Value>High priority</Value>
      </UserDefinedField>
   </Applicant>
</Application>
```

Figure 17: A string user-defined field

Table 12: Data Method Example for Figure 17

Item	Value		
Data Method Name	dm_Obj_UDF_GetStringABC		
Context (Path+Element)	Application.Applicant		
Result Type	string		
Input Variables	none		
Local Variables	**Name**	**Data Type**	**Initial Value**
	xString	string	null

Expression	result := null;
	for every Application.Applicant.UserDefinedField relative to Applicant do
	  if ((Name = "ABC") and (DataType = "string")) then
	    xString := Value;
	  endif
	endfor
	result := xString		

```
<Application>
   <Applicant>
      <UserDefinedField>
         <Name>DEF</Name>
         <DataType>numeric</DataType>
         <Value>22.35</Value>
      </UserDefinedField>
   </Applicant>
</Application>
```

Figure 18: A numeric user-defined field

Table 13: Data Method Example for Figure 18

Item	Value
Data Method Name	dm_Obj_UDF_GetNumericDEF
Context (Path+Element)	Application.Applicant
Result Type	numeric
Input Variables	none
Local Variables	**Name** xString; **Data Type** string; **Initial Value** null

Expression	result := null;
	for every Application.Applicant.UserDefinedField relative to Applicant do
	    if ((Name = "DEF") and (DataType = "numeric")) then
	      xString := Value;
	    endif
	endfor
	if (xString <> null) then result := Val(xString); endif

```
<Application>
   <Applicant>
      <UserDefinedField>
         <Name>GHI</Name>
         <DataType>timestamp</DataType>
         <Value>2005-10-17T13:17:54"</Value>
      </UserDefinedField>
   </Applicant>
</Application>
```

Figure 19: A timestamp user-defined field

Table 14: Data Method Example for Figure 19

Item	Value
Data Method Name	dm_Obj_UDF_GetTimestampGHI
Context (Path+Element)	Application.Applicant
Result Type	timestamp
Input Variables	none
Local Variables	**Name** xString; **Data Type** string; **Initial Value** null

Expression	result := null;
	for every Application.Applicant.UserDefinedField relative to Applicant do
	    if ((Name = "GHI") and (DataType = "timestamp")) then
	      xString := Value;
	    endif
	endfor
	if (xString <> null) then result := TimestampFromStr(xString); endif

```
<Application>
   <Applicant>
      <UserDefinedField>
         <Name>JKL</Name>
         <DataType>date</DataType>
         <Value>2005-10-27</Value>
      </UserDefinedField>
   </Applicant>
</Application>
```

Figure 20: A date user-defined field

Table 15: Data Method Example for Figure 20

Item	Value		
Data Method Name	dm_Obj_UDF_GetDateJKL		
Context (Path+Element)	Application.Applicant		
Result Type	date		
Input Variables	none		
Local Variables	**Name**	**Data Type**	**Initial Value**
	xString	string	null

Expression	result := null;
	for every Application.Applicant.UserDefinedField relative to Applicant do
	  if ((Name = "JKL") and (DataType = "date")) then
	    xString := Value;
	  endif
	endfor
	if (xString <> null) then result := TimestampToDate(TimestampFromStr(xString)); endif		

```
<Application> - Removed for now, as there is no DurationFromStr function.
   <Applicant>
      <UserDefinedField>
         <Name>MNO</Name>
         <DataType>duration</DataType>
         <Value>5 months</Value>
      </UserDefinedField>
   </Applicant>
</Application>
```

Figure 21: A duration user-defined field

Table 16: Data Method Example for Figure 21

Item	Value
Data Method Name	dm_Obj_UDF_GetDurationMNO
Context (Path+Element)	Application.Applicant
Result Type	duration
Input Variables	none

Table 16: Data Method Example for Figure 21 (continued)

Item	Value		
Local Variables	**Name**	**Data Type**	**Initial Value**
	xString	string	null

Expression	result := null;
	for every Application.Applicant.UserDefinedField relative to Applicant do
	  if ((Name = "MNO") and (DataType = "duration")) then
	    xString := Value;
	  endif
	endfor
	if (xString <> null) then result := DurationFromStr(xString); endif		

Using User-Defined Fields in Data Object References

Tables 17 through 19 show examples of data object references that work with user-defined fields. Note that data object references currently cannot use parameters.

Table 17: Data Object Reference Example 1—Application UDF

Item	Value		
DOR Name	dor_Root_UDF_XYZ		
Context (Path+Element)	Application.UserDefinedField		
Conditions	**Field**	**Operator**	**Value | Parameter**
	Name	=	“XYZ”
	DataType	=	“String”
Description	A simple data object reference that returns all Application UserDefinedField objects with a name (and type).		
Issues	This data object reference should not be used if multiple UserDefinedFields exist with the same key data.		

Table 18: Data Object Reference Example 2—Applicant UDF

Item	Value		
DOR Name	dor_Applicant_UDF_XYZ		
Context (Path+Element)	Application.Applicant.UserDefinedField		
Conditions	**Field**	**Operator**	**Value | Parameter**
	Name	=	“XYZ”
	DataType	=	“String”
Description	A simple data object reference that returns all Application.Applicant. UserDefinedField objects with a name (and type).		
Issues	This data object reference should not be used if multiple UserDefinedFields exist with the same key data, or if there are multiple applicant objects, each of which could have a UserDefinedField with the same key.		

Table 19: Data Object Reference Example 2—Primary Applicant UDF

Item	Value		
DOR Name	dor_Applicant_PrimaryUDF_XYZ		
Context (Path+Element)	Application.Applicant.UserDefinedField		
Conditions	**Field**	**Operator**	**Value | Parameter**
	Name	=	"XYZ"
	DataType	=	"String"
	Applicant.ApplicantType=		"Primary"
Description	A simple data object reference that returns all Application.Applicant UserDefinedField objects with a name (and type).		
Issues	This data object reference should not be used if multiple UserDefinedFields exist with the same key data. It includes hardcoded conditions to check for the Primary applicant. This data object reference would not be reusable for other applicant types, but could be copied and edited for each unique type.		

Using User-Defined Fields in Rules

When trying to use user-defined field data in *rule conditions*, keep in mind that it will *always* be a good choice to write a data method to access the user-defined field value. You can ask the following questions to identify whether the user-defined field can be used directly or if a data method or data object reference will be needed:

- Will more than one user-defined field be needed? If so, then data methods are the only choice to access the user-defined field values.
- Does DataType equal "string" for the user-defined field? If so, then the value can be used directly by the rule condition. Otherwise, you will need to use a data method to convert the Value field (which is always a string) into the desired data type.
- Is a data object reference being used? If so, you can create a new data object reference (based on the old one) that goes down one more level to the user-defined field, and include in that data object reference all the conditions in the original data object reference plus conditions on the user-defined field key fields.
- Is this a complex rule? If the conditions of the rule are complex, it is easiest to use a data method to access the user-defined field value.
- Does this rule fire on *any* condition? If so, then a data method is the only choice.

Using User-Defined Fields in Mappings

Given the fact that a mapping needs to be very specific as to which instance of an object is being referenced, if you cannot write the data object reference for the user-defined field, then a data method must be used.

User-Defined Fields Created or Populated by Decision Accelerator

In order for Decision Accelerator to create user-defined fields, there must be a pre-existing agreement between the calling system owners and the Decision Accelerator owners as to the user-defined field location, name, data type, and possible values.

Table 20 illustrates a data method that sets the value of a UserDefinedField child of the Applicant object to the value of the iValue input variable, if the UserDefinedField has a Name value of "Code1".

Table 20: Data Method Example 1—Set String Code1

Item	Value		
Data Method Name	dm_App_UDF_SetCode1		
Context (Path+Element)	Application.Applicant		
Result Type	string		
Input Variables	**Name**	**Data Type**	**Initial Value**
	iValue	string	n/a
Local Variables	**Name**	**Data Type**	**Initial Value**
	xUDF	UserDefinedField	null
Expression	see below		

```
result := null;
if (iValue <> null) then
    // check for existing UDF with the key data.
    for every Application.Applicant.UserDefinedField relative to Applicant do
      if ( ( Name = "Code1") and ( DataType = "string") )
      then
        xUDF := UserDefinedField;
      endif
    endfor
    if (xUDF = null) then
      xUDF := New UserDefinedField;
      xUDF.Value := iValue;
      Attach (Applicant, xUDF)
    else
      xUDF.Value := iValue;
    endif
endif
```

Table 21 illustrates a data method that creates a new user-defined field as a child of the Application.Applicant object. The data method illustrated in Table 22 calls the data method in Table 21 to set the value of the user-defined field with the value generated by a third data method. The Name (ScoringTimestamp) and DataType (timestamp) of the UserDefinedField are passed as parameters.

Table 21: Data Method Example 2—Set String XYZ by Name and Type

Item	Value		
Data Method Name	dm_App_UDF_SetXYZ_ByNameType		
Context (Path+Element)	Application.Applicant		
Result Type	string		
Input Variables	**Name**	**Data Type**	**Initial Value**
	iName	string	n/a
	iDataType	string	n/a
	iValue	string	n/a

Table 21: Data Method Example 2—Set String XYZ by Name and Type (continued)

Item	Value		
Local Variables	**Name**	**Data Type**	**Initial Value**
	xUDF	UserDefinedField	null

Expression	
	```
	result := null;
	if (iValue <> null) then
	  // check for existing UDF with the key data.
	  for every Application.Applicant.UserDefinedField relative to Applicant do
	    if ( ( Name = iName) and ( DataType = iDataType)    ) then
	      xUDF := UserDefinedField;
	    endif
	  endfor
	  if (xUDF = null) then
	    xUDF := New UserDefinedField;
	    xUDF.Name := iName;
	    xUDF.DataType := iDataType;
	    xUDF.Value := iValue;
	    Attach (Applicant, xUDF)
	  else
	    xUDF.Value := iValue;
	  endif
	endif
	```

Table 22: Data Method Example 3—Set a Specific UDF

Item	Value		
Data Method Name	dm_App_SetSpecificUDF		
Context (Path+Element)	Application.Applicant		
Result Type	string		
Input Variables	none		
Local Variables	**Name**	**Data Type**	**Initial Value**
	xValue	string	null

Expression	
	```
	result := null;
	/* Assume we want to set a UDF with a Name = "ScoringTimestamp", a
	DataType of timestamp, and a value from some other method
	dm_App_ScoringTime */

	xValue := TimestampFormat(dm_AppScoringTime,”yyyy-MM-
	dd'T'HH:mm:ss”);
	if (xValue <> null) then
	  dm_App_UDF_SetXYZ_ByNameType( "ScoringTimestamp", "timestamp",
	  xValue);
	endif
	```

Using Data Object References to set User-Defined Fields

If the user-defined field object needs to be created, there is no value in using a data object reference to set its value. The data object reference could only be used to reset the value. As long as a data method was used to create the object, it would make more sense to set the value using the data method.

Using Mappings to set User-Defined Fields

A user-defined field value cannot be set directly from a table mapping in the SMW.

Using Data Method Sequences to set User-Defined Fields

Sequences cannot pass a value parameter to a data method, so then they cannot set a user-defined field without calling a method that creates and sets the user-defined field.

Resubmitting Transactions

To indicate that a transaction is a resubmission, you must enter a value of "true" for the Application/@ResubmissionFlag. To determine which XML nodes can be passed as resubmission input, refer to the **DA Resub In** and **DA+<REMOVED> Resub In** columns of the data model spreadsheet. For example, for a regular Decision Accelerator transaction, the DecisionResponse node is output only. For a Decision Accelerator resubmission transaction, the DecisionResponse node can be submitted on input.

Resubmitted applications can contain new data, such as consumer credit report data retrieved during the first pass. Decision Accelerator does not make any assumptions about resubmitted data; risk management analysts must configure the decision flow(s) and the data methods used in the decision flow(s) to handle the resubmitted data according to your own requirements. The decision flows delivered with Decision Accelerator contain *steps* that handle requests for external data (from <REMOVED> or the FICO® Expansion score service, for example). Because a resubmission transaction could already include such data, the decision flows contain steps that check to see if the data already exists; if it does, the decision flow does not re-request it.

When using a *champion decision flow*/*challenger decision flow*, you should submit the StrategySelectionRandomNumber value from the first pass, to route the application down the same decision flow. If the calling system submits any products in the DecisionResponse/Product node of a resubmission, Decision Accelerator sets their ProductStatusIndicator to "Old" to indicate that, upon return, they were not created in the latest call to Decision Accelerator. Decision Accelerator will ignore any products that are marked as "Old".

Resubmissions can facilitate requests for multiple products. If you want to process applications against all products configured in Decision Accelerator, you cannot use the cross-sell feature, because the product decision must always be "Approve" for a cross-sell to be invoked. Instead, you can send in an application multiple times, to run against each product. You can send in additional data gathered during one of the original submissions, such as consumer credit report data, to avoid pulling it again when running against the rest of the products.

Internal Services

Internal services let you enhance Decision Accelerator functionality with Fair Isaac models and other business logic. Before calling systems can submit transactions requiring internal service data to Decision Accelerator, SMW users must configure decision flows to access the desired internal service. SMW users must provide calling system developers with information about input data required and returned by the internal service.

To use an internal service in a decision flow, SMW users must follow these steps

1 Implement the Blaze Advisor internal service project so that it is accessible to Decision Accelerator, along with any data transformations that may be required.

2 Create a definition for this internal service in the **Internal Service Properties Page**.

3 Create data methods that prepare an internal service request object, invoke the internal service, and then handle the response.

4 Create a data method sequence to execute these data methods.

5 In a decision flow, create a *decision flow step* whose *step implementation* is the data method sequence.

6 Use this decision flow in a product.

Pre-defined Product Strategy Internal Service Requests

Some pre-defined product strategies delivered with Decision Accelerator contain a pre-configured internal service request for <REMOVED> data. The strategies use pre-configured data methods to construct the request to the <REMOVED> project; they also use pre-configured data transformations. If you want to invoke other types of projects, you will need to create new data methods and internal service calls based on the delivered data methods and internal service call.

Important The <REMOVED> project called from the UsedCarLoanStrategy does not contain an actual <REMOVED> model. It *must not* be used to produce <REMOVED>s on applicants in a development or production environment. It is a placeholder for the actual <REMOVED> project, which must be customized for each client. Once an actual <REMOVED> project is installed, a decision flow author will need to modify the internal service that calls it and modify the data method that invokes the internal service. For more information, contact your Fair Isaac account representative.

Adding New Internal Service Projects

At this time, Decision Accelerator can only support *<REMOVED>s* that conform to the following:

- The project must have an entry point that accepts a string and returns a string.
- The string received by the entry point must be one of the following:
  - An XML document whose schema matches the Decision Accelerator XML schema or a subset of the Decision Accelerator XML schema.
  - An XML document or other string derivable from executing an XSL transformation on the above type of document.

  Similar constraints apply to the output argument. The output transformation does not require the XML document sent to the entry point, just the response.

- The project must be deployed using the Blaze Advisor QuickDeployer, using the Java deployment option.
- The deployment's .server file must already be installed in the Services installation folder.

If transformations are applied to the input that Decision Accelerator submits, the resulting XML must conform to the data format expected by the internal service; likewise, transformations applied to the output of the internal service must accommodate the data format produced by the internal service. The transformation files must be placed in the Services installation folder, as noted in the "Web and Internal Services" chapter of the *Capstone Decision Accelerator User's Guide*.

Important When you deploy an internal service project, Blaze Advisor writes the absolute or relative locations of the associated files (such as .adb, class, .jar, or repository files) to the .server file. When it calls the internal service, Decision Accelerator attempts to access the files based on the locations specified in the .server file. You must ensure that each associated file has been installed in the correct location—this may be directly in the Services folder, in a subfolder, or in any other location specified when using the QuickDeployer.

Web Services

Web services allow Decision Accelerator to access various sources of external data. Before calling systems can submit transactions requiring Web service data to Decision Accelerator, SMW users must configure decision flows to access the desired Web service. SMW users must provide calling system developers with information about input data required and returned by the Web service.

To use a Web service in a decision flow, SMW users must follow these steps

1. Obtain connection information for the Web service, along with any data transformations that may be required.
2. Create a definition for this Web service in the **Web Service Properties Page**.
3. Create data methods that prepare a Web service request object, invoke the Web service, and then handle the response.
4. Create a data method sequence to execute these data methods.
5. In a decision flow, create a decision flow step whose implementation is the data method sequence.
6. Use this decision flow in a product.

Pre-defined Product Strategy Web Service Calls

Decision Accelerator can submit multiple Intelligent Data Manager data service requests during a single transaction. Some pre-defined product strategies delivered with Decision Accelerator use pre-configured Web service requests to Intelligent Data Manager to obtain consumer credit report (<REMOVED>) data and FICO® Expansion score data. The strategies use pre-configured data methods to construct data service request objects and submit the data service request to Intelligent Data Manager; they also use pre-configured data transformations. If you want to submit requests for other types of data from Intelligent Data Manager, you will need to create new data methods and Web service calls based on the delivered data methods and Web service calls.

Adding New External Web Services

Decision Accelerator can support both synchronous and asynchronous Web service calls. The Web service must accept a single string argument which must be one of the following:

- An XML document whose schema matches the Decision Accelerator XML schema or a subset of the Decision Accelerator XML schema.
- An XML document or other string derivable from executing an XSL transformation on the above type of document.

Similar constraints apply to the output argument. The output transformation may involve just the Web service's response, or a "merge" of the Web service's response and the original request. For asynchronous calls, Decision Accelerator does not wait for or process any response that is returned from the Web service.

If transformations are applied to the input that Decision Accelerator submits, the resulting XML must conform to the data format expected by the Web service; likewise, transformations applied to the output of the Web service must accommodate the data format produced by the Web service and must conform to the data format expected by Decision Accelerator. The transformation files must be placed in the services installation folder, as noted in the "Web and Internal Services" chapter of the *Capstone Decision Accelerator User's Guide*.

Updating the Data Repository with Final Decision and Offer Data

At runtime, the Process Server sends XML transaction data to the DSS. The FinalApplicationDecision is the final application decision of the calling system (as opposed to the application decision recommended by Decision Accelerator). This data is not provided by Decision Accelerator and must be provided by the calling system.

At runtime, the Process Server sends XML transaction data to the DSS. The following types of data are not provided by Decision Accelerator and must be provided by the calling system:

- **Final Application Decision**—The final application decision of the calling system (as opposed to the application decision recommended by Decision Accelerator).
- **Final Offer History**—The actual offer made to the credit applicant (as opposed to the offer recommended by Decision Accelerator).

You may wish to add this data to the data repository used by the DSS for use in reports according to your own requirements. This section contains some sample SQL queries that you can modify and use to update the data repository reporting tables.

Final Application Decision

Each application is recorded by the DSS each time it is submitted for processing. The data repository will contain a timestamped DATransaction record and a number of DecisionFlowHistory records for each time an application is submitted.

One option is to update the DecisionFlowHistory for the decision associated with the *latest* transaction processed for the application:

```
UPDATE decisionflowhistory1 AS target
   INNER JOIN (
   SELECT tran. hncid, dfh. hncid AS dfhid, superid
   FROM datransaction1 AS tran
      inner join decisionflowhistory1 AS dfh ON tran.hncid = dfh.superid
   WHERE tran.SystemId = "sysid"
      AND tran.ApplicationCrossReferenceId = "APPREF"
      AND dfh.ProductCode = "PRODUCT"
      AND dfh.DecisionFlowName = "DECISONFLOW"
      AND dfh.ProductCategory = "CATEGORY"
   ORDER BY ProcessingTimestamp DESC
   LIMIT 1
   ) AS source ON source.dfhid = target.hncid AND source.superid = target.superid
   SET FinalDecisionResult = "XXXXXX";
```

You may prefer to update the final decision for all occurrences of the DecisionFlowHistory:

```
UPDATE decisionflowhistory1 INNER JOIN datransaction1 ON datransaction1.hncid =
decisionflowhistory1.superid
   SET FinalDecisionResult = "XXXXX"
   WHERE datransaction1.SystemId = "sysid"
      AND datransaction1.ApplicationCrossReferenceId = "REF"
      AND decisionflowhistory1.ProductCode = "PROD"
      AND decisionflowhistory1.DecisionFlowName = "DECISIONFLOW"
      AND decisionflowhistory1.ProductCategory = "CATEGORY";
```

✓ **Note** Substitute values for SystemId, ApplicationCrossReferenceId, ProductCode, ProductCategory, FinalDecisionResult, and DecisionFlowName as appropriate.

Using Decision Accelerator with Intelligent Data Manager

Some nodes in the IBS data model apply only to Decision Accelerator. For example, the DecisionResponse output node is not applicable to Intelligent Data Manager-only transactions, because Intelligent Data Manager does not perform decisioning. Other nodes apply only to Intelligent Data Manager. For example, for non-resubmission input, the <REMOVED>DataServiceRequest, BbDataServiceRequest, FesDataServiceRequest, and SbfeDataServiceRequest children of ExternalDataSource are only used when the calling system submits transactions directly to Intelligent Data Manager.

Some nodes apply only to situations where Decision Accelerator and Intelligent Data Manager are used in conjunction with each other. For example, for transactions in which Decision Accelerator makes a request to Intelligent Data Manager for consumer credit report data, calling systems can submit data to Decision Accelerator in the <REMOVED>RequestDefaults node. Decision Accelerator must be configured to accept this type of data and use it to construct the DataServiceRequest node required by the Intelligent Data Manager Web service.

Request Object Construction

When Decision Accelerator makes are request to Intelligent Data Manager, it must construct and submit the same type of DataServiceRequest object that a calling system would submit if it were calling Intelligent Data Manager directly. For example, for a consumer credit report data request, the DataServiceRequest object contains the following items:

- DataflowName (mandatory)
- RequestedProductType (optional)
- ReportInstructions
- RequestedResponse
- Requested<REMOVED>
- InquiryParameters
- AnalysisParameters
- <REMOVED>Household
- <REMOVED>Applicant

Sources of DataServiceRequest Data

When building decision flows, you can use data methods to populate the DataServiceRequest node. The data model spreadsheet contains information on which DataServiceRequest nodes must be populated for requests to Intelligent Data Manager. The data used to populate the DataServiceRequest node can come from several sources:

1. Areas of input XML documents other than ExternalDataSource. The pre-defined product strategies supplied with Decision Accelerator use data methods that perform this type of task for consumer credit report and FICO® Expansion score requests.
2. The RequestDefault areas of input XML documents (as mentioned above). The pre-defined product strategies supplied with Decision Accelerator *do not* contain examples of data methods that perform this type of task. You will need to add them if you want calling systems to submit data in RequestDefault nodes.
3. Values hard-coded into data methods. If calling systems will not pass data in the request defaults sections of input XML documents, you can use this option. The pre-defined product strategies supplied with Decision Accelerator contain examples of data methods that perform this type of task for consumer credit report and FICO® Expansion score requests. If you want Decision Accelerator to request other types of data from Intelligent Data Manager, you can add data methods to construct request objects using hard-coded values.
4. Configured values in the Intelligent Data Manager Dataflow Configuration Workstation.

See Also For information on using the Intelligent Data Manager Dataflow Configuration Workstation, see the IDM.

More detailed information on Decision Accelerator requests to Intelligent Data Manager is contained in the individual data source chapters.

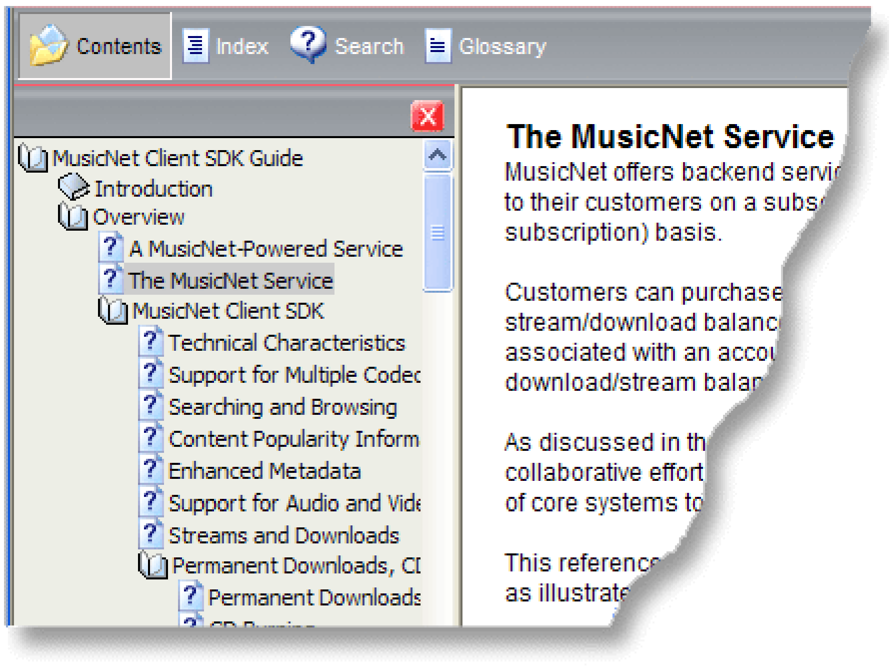
See Also

- For more information on consumer credit report requests and consumer ARF input, see <REMOVED>
- For more information on list of similars requests, business credit report requests, and business ARF input, see <REMOVED>
- For more information on FICO® Expansion score service requests, see <REMOVED>
- For more information on SBFE data requests, see <REMOVED>

Conditional Branching

Decision Accelerator can perform conditional branching based on data from Intelligent Data Manager. For example, some of the pre-defined product strategies delivered with Decision Accelerator check to see whether credit report data exists before requesting it. After credit report data has been requested, a decision flow determines whether full consumer credit report analysis data is available, or whether there is a thin file or no report. If there is a full report, it may be analyzed to see whether it indicates fraud. If the file is thin or non-existent, Decision Accelerator may request FICO® Expansion score data from Intelligent Data Manager.

MusicNet Client SDK Developer's Guide



MusicNet Client SDK Guide
Introduction
Overview
A MusicNet-Powered Service
The MusicNet Service
MusicNet Client SDK
Technical Characteristics
Support for Multiple Codec
Searching and Browsing
Content Popularity Inform
Enhanced Metadata
Support for Audio and Vide
Streams and Downloads
Permanent Downloads, CI
Permanent Downloads

The MusicNet Service

MusicNet offers backend servic
to their customers on a subs
subscription) basis.

Customers can purchase
stream/download balanc
associated with an accou
download/stream balan

As discussed in th
collaborative effort
of core systems to

This referenc
as illustrate

The MusicNet Client SDK

The MusicNet API components encapsulate the functionality needed by a client application to access the MusicNet service and to manage content licensing tasks. It does not include graphical interface elements or attempt to hook into any existing applications. As the client developer, you can use the MusicNet SDK to incorporate MusicNet services into your own client application.

All MusicNet client/server functionality is provided as COM-compliant interfaces. COM functions return HRESULTs. When connecting to a remote server, functions must be able to indicate possible network failure in addition to function-call success or failure. This is accomplished by having all functions return an HRESULT. If a function must return a value outside the scope of an HRESULT, an output parameter is used. For error code details, see "Errors and Error Handling" on page 939.

The MusicNet SDK is organized into interfaces that manage specific features, such as logging in or searching. Most MusicNet SDK features can be implemented using a common template:

- Create an object exposing the functionality of interest.
- Set the necessary properties.
- Create a listener for an anticipated event.
- Call the Execute method for the object.
- Handle the subsequent event.

All communication with MusicNet is asynchronous. The client developer must use connection points to listen for events.

✓ **Note** The sample code shown in this document was written using Visual C++ 6.0.

Updating the DRM

When installing and setting up a MusicNet client for the first time on any machine, the machine's digital rights management (DRM) system will need to be updated to handle MusicNet’s secure content.

MusicNet currently provides content for two different DRMs: RealNetworks MCS, and Microsoft’s Windows Media (WM). There is a separate process for updating each DRM.

RealNetworks Auto Update

A client application must be designed to implement Auto Update in two circumstances:

When setting up on a machine for the first time

In response to an update request event from the media player

The RN Auto Update process is primarily used to get or replace licensing components in cases of upgrades or responses to security breaches. On an already installed client, the Auto Update process is triggered when a consumer attempts to play content. A consumer will not be able to play content or acquire new licenses until the Auto Update process has been successfully completed.

Coding Issues

Initiate Auto Update during initial setup, or in response to the event **_IMNTLCEvents::OnRequestUpdate** when attempting to play content.

Create an **IMNProductInfo** object and set all properties as specified in the API.

Create an **IMNAutoUpdate** object and call **IMNAutoUpdate::Initialize**, referencing the **IMNProductInfo** object, to validate the product information supplied.

Before initiating Auto Update, unload the TLC. Without taking this step, consumers will need to reboot after Auto Update is completed.

Call **IMNAutoUpdate::DoPlugInRequest** to initiate Auto Update. The parameter *fileExts* should be set to the list of components returned with **_IMNTLCEvents::OnRequestUpdate**.

Sample Code

The following code fragment illustrates how to perform Auto Update for the RealNetworks DRM.

```
. . .

// create IMNProductInfo object
```

```
CComPtr<IMNProductInfo> spProductInfo;
HRESULT hr = S_OK;
spProductInfo.CoCreateInstance(__uuidof(MNProductInfo));
if (FAILED(hr))
{
// handle error ...
}
// set IMNProductInfo properties
spProductInfo->put_CompanyName(L”RealNetworks”);
spProductInfo->put_ProductName(L”MyProductName”);
spProductInfo->put_ProductTitle(L”MyCProductTitle”);
spProductInfo->put_VersionString(L”1.1.1.1”);
spProductInfo->put_MajorVersion(1);
spProductInfo->put_MinorVersion(1);
spProductInfo->put_ReleaseNumber(1);
spProductInfo->put_BuildNumber(1);
spProductInfo->put_LangID(L”EN”);
// create IMNAutoUpdate object
CComPtr<IMNAutoUpdate>spAutoUpdate;
HRESULT hr = S_OK;
spAutoUpdate.CoCreateInstance(__uuidof(MNAutoUpdate));
if (FAILED(hr))
{
// handle error ...
}
// validate product information
if (FAILED(spAutoUpdate->Initialize(CComBSTR(pluginFolder), (long)hWnd,
CComBSTR(skinURL),
spProductInfo)))
{
// handle error ...
}
```

```
// begin listening for events from Auto Update
DispEventAdvise(spAutoUpdate, &__uuidof(_IMNAutoUpdateEvents));
// send an Auto Update request
if (FAILED(spAutoUpdate->DoPlugInRequest(CComBSTR(componentList))))
{
// handle error ...
}
```

Windows Media Individualization

To access MusicNet's secure WMA content, the Windows Media DRM must be upgraded, which involves individualizing the DRM to a specific machine. Individualization needs to be done only once per machine.

Windows Media individualization is ideally done during installation and set up, but must be completed before a consumer can access secure WMA content. Individualization can be done in response to an event from a media player.

Coding Issues

Create an **IMNIndividualization** object and call Individualize. The *fileUrl* parameter may reference the content file the consumer is attempting to play, or it may reference a URL provided by MusicNet for individualization at setup. See the Partner Integration Handbook for more information about this file.

Individualize will return two events: **OnBegin** and **OnEnd**. **OnEnd** provides a flag to indicate success or failure and several reasons for failure.

Sample Code

The following code fragment illustrates how to perform individualization for the Windows Media DRM.

```
// create IMNIndividualization object
CComPtr<IMNIndividualization>spIndividualization;
HRESULT hr = S_OK;
spIndividualization.CoCreateInstance(__uuidof(IMNIndividualization));
if (FAILED(hr))
{
// handle error ...
}
// begin listening for events from Individualization
DispEventAdvise(spIndividualization, &__uuidof(_IMNIndividualizationEvents));
// initiate individualization process
if (FAILED(spIndividualization->Individualize(fileURL)))
{
// handle error ...
}
```

Logging In

Once you've created an **IMNUserSession** object and set the necessary properties, create an **IMNLogin** object.

There are three required properties in the **IMNLogin** interface:

- *RetailerID*
  MusicNet-assigned identifier. This number is a constant and is included in the Partner Integration Package.
- *RetailerToken*
  Used to verify authenticity of the login. Partners are responsible for establishing a mechanism to create retailer tokens, communicate them through the **IMNLogin** interface, and then authenticate them during the login process (for login authentication, see the MusicNet E-commerce SDK Programmer's Reference).
- ClientGUID
  Stored locally in the subscriber profile, identifies the unique combination of subscriber, machine, and retailer. The client application must implement a way to generate ClientGUIDs, using the standard COM/OLE format {8-4-4-4-12} with curly brackets and values between 1-9, a-f, and A-F.

- **ClientGUIDs** are tracked on the MusicNet server with the customer account information; they are used to identify subscribers and to determine which machines are download enabled for an account.
- On successful completion of a login, the SDK returns the download activation status and the **SubscriptionGUID** assigned to the account. The **SubscriptionGUID** is used to check the remaining time before the subscription license expires and to get license information from the DRM.
- If the subscription is not valid (that is: cancelled, deactivated, suspended, or closed), the login attempt will fail.
- The **DownloadActivationStatus** indicates whether the machine currently in use is download-enabled. If the status is "active" or "activable", the subscriber can download and stream content to that machine. If the status is "inactive" the subscriber can stream content and get permanent downloads only.
- **MachineID Tags**
  Eventually, we plan to use MachineID tags instead of randomly generated ClientGuids to identify machines at login. This practice will improve security because MachineIDs are based on MAC addresses and HDD serial numbers, so they cannot be transferred to other machines.
- Create an **IMNLogin** object using the **IMNUserSession::CreateLogin** method. Once you've created the login object, set the **RetailerID**, **RetailerToken**, and **ClientGUID** properties; all three are required to connect to the MusicNet service. Call **Execute** to initiate a login request.
- For handling concurrent logins, rather than automatically terminating the first session, you can choose to protect the first session and not allow a second login. To do this, change the **IMNLogin::AutoLogout** property to FALSE.

- There is a security check with each login confirming that the client is actually contacting the MusicNet service. If the security check fails, the SDK will return the error 0x800C000E. In this event, display a message to the subscriber with language such as “Server security access failure. There was a security problem when accessing the server. Please contact customer support.”

- Once a user session has been established, communication with MusicNet becomes asynchronous. The client developer must use connection points to listen for events.

Sample Code

The following code fragment illustrates how to create and execute a login object and how to establish a listener.

```
. . .
// create IMNLogin object
// spUserSession previously created and configured
CComPtr<IMNLogin> spLoginCmd;
if (FAILED(spUserSession->CreateLogin(&spLoginCmd)));
{
// handle error ...
}
// set properties
spLoginCmd->put_RetailerToken(token);
spLoginCmd->put_RetailerId(retailerCode);
spLoginCmd->put_ClientGUID(userGUID);
// begin listening for events from spLoginCmd
DispEventAdvise(spLoginCmd, &__uuidof(_IMNEvents));
// send a login request
if (FAILED(spLoginCmd->Execute()))
{
// handle error ...
}
```

Acquiring and Managing Windows Media Licenses

In order for users of Windows Media Player (WMP) to stream music, to play downloads, to transfer downloads to portable devices, or to burn permanent downloads to CDs, they must acquire the following licenses:

Subscription licenses

Subscriptions can vary and may include stream rights, subscription download rights, or both stream and subscription download rights. Subscriptions are automatically renewed at the end of each subscription period.

Stream licenses

Allow users to play streaming content. For both WMP9 and WMP10 license styles, users obtain one stream license that lasts for 30 days from the moment it's issued and applies to all stream files played during that period; one license works with multiple files because all streamed content is encrypted with the same key.

Subscription download licenses

Depending on their versions of Windows Media Player, subscribers will use of the two styles of subscription download licenses that are available:

WMP10 license style

To play a subscription download, subscribers using WMP10 need two licenses:

Root license

A group license that enables subscribers to play all files (with valid leaf licenses) during the root license period. Root licenses last for 33 days from the moment they're issued. All client applications using the MusicNet SDK should try to renew root licenses before they expire. The 33-day duration provides an extra three days as a buffer, so the client application can renew a root license before it expires. The WMAquirePlayRights interface enables clients to renew root licenses if there are less than 3 days left before they expire.

Note: Root licenses can be requested as often as at each login, but this may produce unwanted consequences:

Unwanted server load (Business Logic server, DB, and License server).

Possible client side performance issues. This is untested with this version of the DRM, but in previous versions of the DRM, scalability issues surfaced when too many licenses were registered.

Leaf license

Each downloaded file has its own leaf license associated with it, which gives users who have a valid root (group) license the right to play that specific file. Leaf licenses never expire, so each subscription period, users must only renew their one group license to continue listening to any number of leaf licensed files; this method makes license renewal more efficient since it's unnecessary to renew large numbers of individually license files.

WMP9 license style

Enabling Portable Device Metering in Applications

To enable device metering, your application must call **IMNUserSession::WMDeviceAvailable**. Call this method only if a user:

- Has a portable subscription;
- Uses Windows Media Player 10;
- Is logged into the MusicNet service;
- Has a Janus device connected to the PC;
- Is likely to keep the device connected long enough to complete this process, which is most likely prior to device synchronization or when the device is idle.

Note: The metering process may increase synchronization time.

Parameter

As a parameter to **WMDeviceAvailable,** the client application should provide the device name or the device canonical name of the portable device. If the application does not provide a device name (empty string), the SDK will attempt to collect metering data from the first device that connects to the PC.

It's unclear if all portable devices can handle concurrent operations, so we recommend that you wait to perform other operations until this process is complete.

The client application can listen to the **_IMNDeviceEvents** interface to determine when device meter reporting is successfully completed.

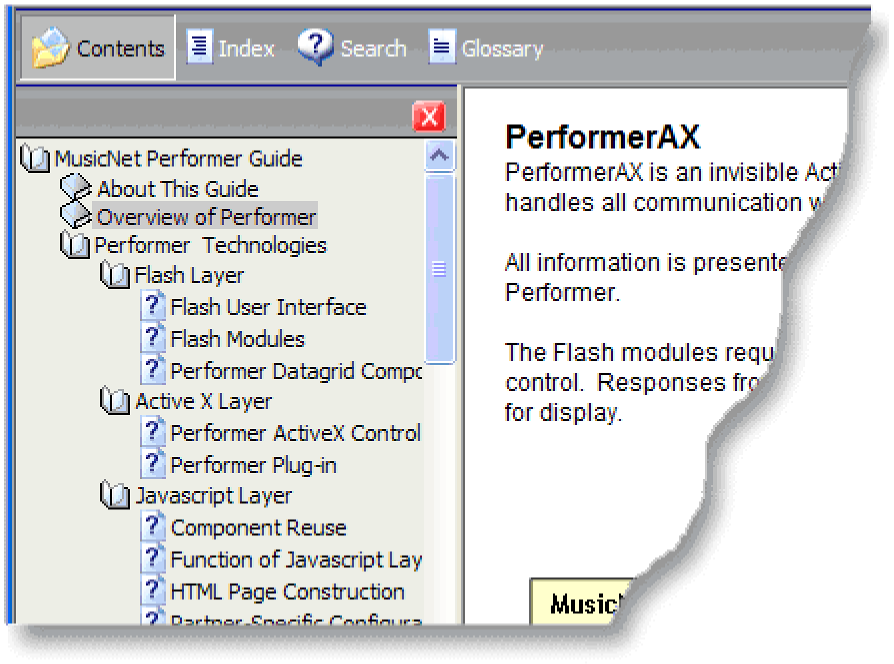
The application can cancel the meter reporting process by calling **IMNUserSession::WMDeviceUnavailable.**

There can be only one meter reporting operation at any give time.

Sample Code

```
//CDeviceManager is the class that handles the communication with devices

//Call this method right before a transfer to device happens or when a device is connected for

// a long period of time with no activity

HRESULT CDeviceManager::NotifyDeviceAvailable(BSTR bstrDeviceName)

{

//The user must be login in

//If the application can do autologin it should do so to ensure device meters are reported

if (!signedIn_)

{

CLog::Log("The user is not signed in. Device metering process stopped.");
```

MusicNet Performer Developer’s Guide



MusicNet Performer Guide
About This Guide
Overview of Performer
Performer Technologies
Flash Layer
Flash User Interface
Flash Modules
Performer Datagrid Compo
Active X Layer
Performer ActiveX Control
Performer Plug-in
Javascript Layer
Component Reuse
Function of Javascript Lay
HTML Page Construction

PerformerAX

PerformerAX is an invisible Act
handles all communication w

All information is presente
Performer.

The Flash modules requ
control. Responses fro
for display.

Music

Flash Modules

Over 50 Flash modules are embedded in the HTML pages of Performer; different pages display different combinations of these modules. Depending on their function, modules accept certain types of data and render specific user interfaces based on that data.

FSCommands

The Flash modules issue FSCommands in response to user actions. FSCommands contain a name and a value; this data is interpreted by the Javascript layer and used to call methods from PerformerAX.

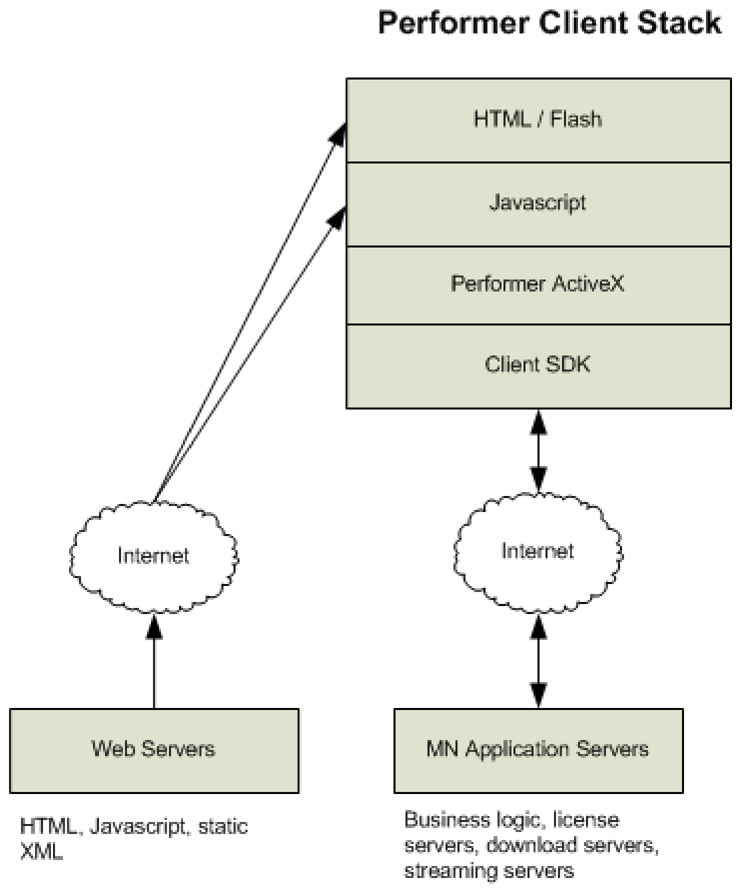
PerformerAX returns responses through the Javascript layer; the javascript layer interprets the responses and routes them to the correct Flash modules to display.

setVariable Commands

Javascript triggers “setVariable” commands in order to pass data to the Flash modules. The Flash modules use a mechanism called a "watch" that enables the modules to automatically change their user interface according to the incoming setVariable data.

Parsing of XML Data

Javascript usually does not parse XML data; instead, it sends XML data returned by PerformerAX directly to the Flash modules for interpretation.



FSCommands and Arguments

Flash modules issue FSCommands in response to user actions. FSCommands contain a name and a value; this data is interpreted by the Javascript layer and used to call methods from PerformerAX.

PerformerAX returns responses through the Javascript layer; the Javascript layer interprets the responses and routes them to the correct Flash modules to display.

MusicNet Standard DRM Rights Developer’s Guide

Windows Media DRM 10 Rights

To play a subscription download, users of WMDRM 10 need a leaf license and a root license, which form a license chain that unlocks the protected content. The root license is tied to the PC or to the portable device and carries a set of rights that unlock the leaf licenses. Each leaf license is tied to the root license and uses its own set of rights to unlock the downloaded file. This section describes the rights associated with each type of license.

Leaf Licenses

Each subscription download has a leaf license associated with it, which gives users who have a valid root (group) license the right to play or to copy that specific file. Leaf licenses never expire, so each subscription period, users must only renew their one root license to continue listening to any number of leaf licensed files.

The leaf license for each subscription download carries a set of rights (summarized below) that permit a subscriber to both copy and play the file.

Root Licenses

A root license enables a subscriber to play all files (with valid leaf licenses) during the root license period. Root (group) licenses last for 33 days from the moment they're issued, which overlaps with the date of the next license update. Client applications should renew root licenses before they expire. The 33-day duration provides an extra three days as a buffer, so the client application can renew a root license before it expires.

Root License Priority

Each root license we deliver has a priority property (http://msdn2.microsoft.com/en-us/library/ms985940.aspx) assigned to it, which determines the order in which it is consumed when an older (but still valid) root license is already in place.

Root license priority will increase over time to allow new root licenses to be immediately copied to a device when it is synchronized.   The practice of using priority to override older root licenses with newer ones enables content to be available for users to playback on a device for as long as possible.

Use Case

Client application downloads a set of tracks, each with it's own leaf license. The client also downloads one root license that expires in 33 days.

Client application transfers the tracks (and licenses) to a portable device. The tracks will play for 33 days on the device until the user's root license expires.

After 30 days the user opens the client application.   The client automatically gets a new root license, which is valid for another 33 days.

When the user synchronizes or transfers new tracks to the (same) device, a newer root license (with a higher priority than the old one) is also transferred to the device. The newer license with the higher priority overrides the old one, so the user will have another 33 days of playback available on the device.

In the past, root license priority was a constant, so the newer license would have had the same priority as the old one. The problem with this practice was that the new root license would not be copied to the device during synchronization since the old one was still valid. This means that after step 4 the content would play on the device for only three more days. Now a new root license will always have a higher priority than an old one, so the new license will always be copied to the device and the content will play on the device for 33 more days, which provides a better user experience.

Code Samples

This section contains codes samples in both Java and C#.

- Java Code Samples
- C# Code Samples

Java Code Samples

I have used both Eclipse and Netbeans to write my Java code samples.

disambiguateTerms

This code sample demonstrates how to retrieve a list of disambiguated terms from the xPatterns platform.

JAVA Example Code -- Disambiguate Terms
  This example demonstrates how to retrieve a list of disambiguated terms.

```
package com.xrelevance.xpatterns_text.text;

  import java.util.Arrays;

  //Required library: xpatterns-api-model-3.0.0.jar
  //Download and add the jar as a dependency to your project.  This
  //library provides HttpEntity, HttpHeaders, HttpMethod, xPatterns exceptions,
  // and the xPatternsRestTemplate.  It also provides a wrapper for the xPatterns disambiguation
  // APIs.
  import org.springframework.http.HttpEntity;
  import org.springframework.http.HttpHeaders;
  import org.springframework.http.HttpMethod;
  import org.springframework.web.client.xPatternsRestTemplate;

  import com.xpatterns.xPatternsRestClientException;
  import com.xpatterns.xrelevance.api.model.Disambiguation;

  public class DisambiguateTermsExample {
    public static void main(String[] args) {

        // Wiki1k names an example domain expert that will provide disambiguation
        // content.
        String expertId = "wiki1k";
        // Your Atigeo representative will provide an xPatterns host and serviceRoot.
        String serverUrl = "https://services3.xpatterns.com/";
        String apiController = "xrelevance-api-rest";

        // Defines the current domain expert.
        String methodName = "/relevance/disambiguate/";
        String pathArguments = "expert/" + expertId + "?";
        String termList = "term=Seattle&term=Seahawks";
        int termOffset = 0;
        int termLimit = 10;

        // Assemble the query string.
        String queryString = serverUrl + apiController + methodName + pathArguments
                + termList + "&" + "offset=" + termOffset + "&" + "limit=" + termLimit;

        // Provides a container for the xPatterns API Key.
        HttpEntityrequestEntity = getEntity();

        // This object stores disambiguation results and exposes properties that
        // enable you to get and set disambiguation terms and queryIds.
        Disambiguation disambiguation = null;

        try {
            // All API calls take place via the xPatterns REST client.
            xPatternsRestTemplate client = new xPatternsRestTemplate();
            disambiguation = client.exchange(queryString, HttpMethod.GET,
                    requestEntity, Disambiguation.class).getBody();
        } catch (xPatternsRestClientException e) {
            System.out.println(e.toString());
        }
        // Print disambiguated terms to the console.
        System.out.println(disambiguation.getTerms().toString());

    }

    // Represents the body of the request / response.
    private static HttpEntity getEntity() {
        HttpHeaders requestHeaders = new HttpHeaders();

        // An Atigeo representative will provide login credentials to the
        // xPatterns Admin Portal, where
        // you can manage API keys for each public method.
        requestHeaders.put("apikey",
                Arrays.asList("cde84e18-a0ca-4d0e-b690-a01076de4507"));

        //The warnings being suppressed are a Spring artifact.
        //@SuppressWarnings({ "unchecked", "rawtypes" })
        HttpEntity requestEntity = new HttpEntity(requestHeaders);
        return requestEntity;
    }
}
```

getContent

This code sample demonstrates how to retrieve content that is filtered and sorted by relevance according to specified preferences.

```
//JAVA Example Code -- GetContent
 //This example demonstrates how to retrieve content filtered and sorted by relevance according to specified
preferences.
 package com.xrelevance.xpatterns_text.text;

 import java.util.Arrays;

 //Required library: xpatterns-api-model-3.0.0.jar
 //Download and add the jar as a dependency to your project.  This
 //library provides HttpEntity, HttpHeaders, HttpMethod, xPatterns exceptions,
 // and the xPatternsRestTemplate.  It also provides a wrapper for the xPatterns disambiguation
 // APIs.

 import org.springframework.http.HttpEntity;
 import org.springframework.http.HttpHeaders;
 import org.springframework.http.HttpMethod;
 import org.springframework.web.client.xPatternsRestTemplate;

 import com.xpatterns.xPatternsRestClientException;
 import com.xpatterns.xrelevance.api.model.RelevantContent;

 public class GetContentExampleCode {

     public static void main(String[] args) {

         // Wiki1k names an example content corpus that will provide relevant
         // terms.
```

```
            String contentId = "wiki1k";
            // Wiki1k names an example expert corpus that will provide
            // disambiguation.
            String expertId = "wiki1k";
            String l = "test";

            // Your Atigeo representative will provide an xPatterns serverUrl and an
            // apiController.
            String serverUrl = "http://192.168.12.80:8080";
            String apiController = "/xrelevance-api-rest/relevance";
            int termOffset = 0;
            int termLimit = 10;

            String pathArguments = "/relevantContent/content/" + contentId
                    + "/expert/" + expertId + "?" + "l=football&" + "d=hockey&"
                    + "c=sports&" + "offset=" + termOffset + "&" + "limit="
                    + termLimit;

            // Assemble the query string.
            String queryString = serverUrl + apiController + pathArguments;

            // Provides a container for the xPatterns API Key.
            HttpEntity<?> requestEntity = getEntity();

            // This object stores the url terms and exposes properties that enable
            // you to get and set url terms and relevancy scores.
            RelevantContent relevantContent = null;
            try {
                // All API calls take place via the xPatterns REST client.
                xPatternsRestTemplate client = new xPatternsRestTemplate();
                relevantContent = client.exchange(queryString, HttpMethod.GET,
                        requestEntity, RelevantContent.class).getBody();
            } catch (xPatternsRestClientException e) {
                System.out.println(e.toString());
            } catch (Exception e) {
                System.out.println(e.toString());
            }
            // Print content properties to the console.
            System.out.println(relevantContent.getContent().length);
            System.out.println(relevantContent.getContext().size());
            System.out.println(relevantContent.getDislikes().size());
            System.out.println(relevantContent.getTotalFound().toString());

        }

        private static HttpEntity<?> getEntity() {
            HttpHeaders requestHeaders = new HttpHeaders();

            // An Atigeo representative will provide login credentials to the
            // xPatterns Admin Portal, where you can manage API keys for each public
            // method.

            requestHeaders.put("apikey",
                    Arrays.asList("cde84e18-a0ca-4d0e-b690-a01076de4507"));

            // The warnings being suppressed are a Spring artifact.
            @SuppressWarnings({ "unchecked", "rawtypes" })
            HttpEntity<?> requestEntity = new HttpEntity(requestHeaders);
            return requestEntity;
        }
    }
```

getContentTerms

This code sample demonstrates how to retrieve terms from a content corpus, which are relevant to the specified document.

```
//JAVA Example Code getContentTerms
 // This example demonstrates how retrieve content terms from a corpus, which are relevant to the specified
document.
 package com.xrelevance.xpatterns_text.text;

 import java.util.Arrays;

 //Required library: xpatterns-api-model-3.0.0.jar
 //Download and add the jar as a dependency to your project.  This
 //library provides HttpEntity, HttpHeaders, HttpMethod, xPatterns exceptions,
 // and the xPatternsRestTemplate.  It also provides a wrapper for the xPatterns disambiguation
 // APIs.
 import org.springframework.http.HttpEntity;
 import org.springframework.http.HttpHeaders;
 import org.springframework.http.HttpMethod;
 import org.springframework.web.client.xPatternsRestTemplate;

 import com.xpatterns.xPatternsRestClientException;
 import com.xpatterns.xrelevance.api.model.ContentTerm;

 public class GetContentTermsExample {
     public static void main(String[] args) {

         // Wiki1k names an example content corpus that will provide content
         // terms.
         String contentId = "wiki1k";
         // Your Atigeo representative will provide an xPatterns host and
         // serverUrl and apiController.
         String documentId = "1";
         String serverUrl = "https://services3.xpatterns.com/";
         String apiController = "xrelevance-api-rest/relevance";

         String methodName = "/contentTerms/";
         String pathArguments = "content/" + contentId + "/" + "documentId"
                 + "/" + documentId + "?";
         int termOffset = 0;
         int termLimit = 10;

         // Assemble the query string.
         String queryString = serverUrl + apiController + methodName
                 + pathArguments + "offset=" + termOffset + "&" + "limit="
                 + termLimit;

         // Provides a container for the xPatterns API Key.
         HttpEntity<Object> requestEntity = getEntity();

         // This object stores the content terms and exposes properties that
         // enable you to
         // get and set content terms and relevancy scores.

         ContentTerm contentTerm = null;

         try {
             // All API calls take place via the xPatterns REST client.
             xPatternsRestTemplate client = new xPatternsRestTemplate();
             contentTerm = client.exchange(queryString, HttpMethod.GET,
                    requestEntity, ContentTerm.class).getBody();
        } catch (xPatternsRestClientException e) {
            System.out.println(e.toString());
        }
        // Print content terms and relevancy scores to the console.
        System.out.println(contentTerm.getTerm());
        System.out.println(contentTerm.getScore());

    }

    // Represents the body of the request / response.
    private static HttpEntity getEntity() {
        HttpHeaders requestHeaders = new HttpHeaders();

        // An Atigeo representative will provide login credentials to the
        // xPatterns Admin Portal, where
        // you can manage API keys for each public method.
        requestHeaders.put("apikey",
                Arrays.asList("cde84e18-a0ca-4d0e-b690-a01076de4507"));

        // The warnings being suppressed are a Spring artifact.
        // @SuppressWarnings({ "unchecked", "rawtypes" })
        HttpEntity<?> requestEntity = new HttpEntity<Object>(requestHeaders);
        return requestEntity;
    }
}
```

C# Code Samples

I use Visual Studio to write my C# code samples.

C# Web Services Client

This code sample demonstrates how to use a web services client to make an API call to xPatterns. I also wrote the server-side code for this web service. This service supported a customer demo, which required the demo application to convert a PDF file to a text string prior to submitting the file to xPatterns.

```
PartnerService.PartnerServiceClient mypartnerService = new PartnerServiceClient();
mypartnerService.ClientCredentials.UserName.UserName = "UserNameProvidedByAtigeo";
mypartnerService.ClientCredentials.UserName.Password = "PasswordProvidedByAtigeo";
// set corpusId (Assigned when you upload your corpus to the Partner Portal)
String corpusId = "36";
// set user ID: a user Guid as string
string userId = "36D72A36-6CA8-418B-856E-A7D02B1092B7";
// Create a date filter.
PartnerService.DateFilter dateFilter = new PartnerService.DateFilter();
dateFilter.BeginDate = DateTime.Parse("1/13/2000");
dateFilter.EndDate = DateTime.Parse("10/26/2010");
// Set the relevance filter
double relevanceFilter = 0;

// Get a list of content IDs
PartnerService.ContentByUserResult contentByUserResult = mypartnerService.GetContentByUser(corpusId,
userId, SuggestionsBehavior.IgnoreSuggestions, null, dateFilter, null, relevanceFilter, 0, 100,
ContentSortOrder.Random);
```

PDF to Text Converter

This code sample is a web service client I created to convert PDF files to Text Files, which we used to support a customer demo.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
namespace WebApplication2
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string fileName = "C:/Users/PhilG.LIFE/Desktop/text.pdf";
            prepareFile(fileName);
        }
        /// <summary>
        /// This client calls my PDF to Text File converter, which returns the file as a string.
        /// </summary>
        /// <param name="filename">Pass the file path to Upload Web Service</param>
        private void prepareFile(string fileName)
        {
            try
            {
                // Get the exact file name from the path
                String strFile = System.IO.Path.GetFileName(fileName);
                Byte[] f = new Byte[0];
                com.xpatterns.srvcsb5.FileUploader serviceClient = new com.xpatterns.srvcsb5.FileUploader();
                string convertedFile = serviceClient.UploadFile(f, fileName);
                Console.WriteLine("convertedFile: " + convertedFile.ToString());
                // Get the file information form the selected file
                FileInfo fInfo = new FileInfo(fileName);
                // Get the length of the file to see if it is possible
                // to upload it (with the standard 4 MB limit)
                long numBytes = fInfo.Length;
                double dLen = Convert.ToDouble(fInfo.Length / 1000000);
                // The default limit of the PDF is 4 MB on web server.
                // You can change this limit by editing the web.config, if you want to allow larger uploads.
                if (dLen < 4)
                {
                   // Set up a file stream and binary reader for the
                    // selected file
                    FileStream fStream = new FileStream(fileName, FileMode.Open, FileAccess.Read);
                    BinaryReader br = new BinaryReader(fStream);
                    // Convert the file to a byte array
                    byte[] data = br.ReadBytes((int)numBytes);
                    br.Close();
                    // Pass the byte array (file) and file name to the web service
                    string sTmp = serviceClient.UploadFile(data, strFile);

                    string filePath = "C:\\Users\\PhilG.LIFE\\Documents\\Conversions\\";
                   FileStream fs = new FileStream(filePath + fInfo.Name.Remove(fInfo.Name.Length - 4) + ".txt", FileMode.CreateNew,
FileAccess.Write, FileShare.Write);
                    fs.Close();
                    StreamWriter sw = new StreamWriter(filePath + fInfo.Name.Remove(fInfo.Name.Length - 4) + ".txt", true,
Encoding.ASCII);
                    sw.Write(sTmp);
                    sw.Close();
                    fStream.Close();
                    fStream.Dispose();
                    // This will always say OK unless an error occurs,
                    // If an error occurs, the service will return an error message
                    MessageBox.Show("File Upload Status: " + sTmp, "File Upload");
                }
                else
                {
                    // Display a message if the file was too large to uploadj.
                    MessageBox.Show("The file selected exceeds the size limit for uploads.", "File Size");
                }
            }
            catch (Exception ex)
            {
                // Display an error message to the user
                MessageBox.Show(ex.Message.ToString(), "Upload Error");
            }
        }
    }
}
```

getContent with submitFeedback

The xPatterns platform had a feature that accepts feedback on the quality of content it delivers, which helps to fine tune results. This example demonstrates how it's done.

```
PartnerService.ContentByUser[] GetContentByUser(string userId, PartnerService.ContextFilter contextFilter,
PartnerService.PriceFilter priceFilter, PartnerService.ContentSortOrder contentSortOrder)
    {
        PartnerService.ContentByUser[] contentByUser = null;
        PartnerService.PartnerServiceClient partnerService = null;
        try
        {
            partnerService = new PartnerService.PartnerServiceClient();
            // Set partner user name.
            partnerService.ClientCredentials.UserName.UserName = "admin";
            // Set partner user password.
            partnerService.ClientCredentials.UserName.Password = "admin";
            // Date filter.
            PartnerService.DateFilter dateFilter = null;
            // Relevance filter.
            double relevanceFilter = 0;
           PartnerService.ContentByUserResult contentByUserResult = partnerService.GetContentByUser(corpusId,
userId, SuggestionsBehavior.IgnoreSuggestions, contextFilter, dateFilter, priceFilter, relevanceFilter, 0, 100,
contentSortOrder);
            if (contentByUserResult.Status == 0)
            {
                contentByUser = contentByUserResult.Content;
                QueryId = contentByUserResult.Id;
            }
            else
            {
                // Write error code and message to the console.
                Console.WriteLine("Error code:" + contentByUserResult.Status.ToString());
                Console.WriteLine("Message:" + contentByUserResult.Message);
                contentByUser = null;
            }
        }
        catch (Exception ex)
        {
            // Unhandled error.
            Console.WriteLine(ex.Message);
        }
        finally
        {
            if (partnerService.State == System.ServiceModel.CommunicationState.Opened)
            {
                partnerService.Close();
            }
        }
        return contentByUser;
    }

    private void SubmitFedback(string contentId)
    {
        PartnerService.PartnerServiceClient partnerService = null;
        try
        {
            partnerService = new PartnerService.PartnerServiceClient();
            // Set partner user name.
            partnerService.ClientCredentials.UserName.UserName = "admin";
            // Set partner user password.
            partnerService.ClientCredentials.UserName.Password = "admin";
```

```
                PartnerService.ContentFeedbackResult result = partnerService.SubmitContentFeedback(QueryId,
corpusId, userId, contentId, DateTime.Now, "WebSite");
            }
            catch (Exception ex)
            {
                // Unhadle error.
                Console.WriteLine(ex.Message);
            }
            finally
            {
                if (partnerService.State == System.ServiceModel.CommunicationState.Opened)
                {
                    partnerService.Close();
                }
            }
        }
```

API Reference Guides

This section contains code samples for Web services from several API Reference Guides that I have written over the years. You'll find both SOAP and REST examples here.

- xPatterns API Reference Guide
- IBS Agreement Management Web Service API Reference Guide
- MusicNet CAM API Reference Guide

xPatterns API Reference Guide

disambiguateTerms(POST)

Returns a list of scored terms that can be used to refine the possible meanings of the input term. For example, the term Mercury becomes clearer when disambiguated with such terms as element, planet, or mythology.

Base URL

https://<Removed>/xrelevance-api-rest/relevance

Resource URI

/disambiguate/expert/{expertId}

Path Variables	Description
String expertId	Identifies the domain expert that will be used to disambiguate terms.

Request Body	Description
ExpertDisambiguationModel query	Contains terms to disambiguate.

Returns

Returns a list of disambiguated terms and their relevancy scores. Scores measure the relevancy of each expanded term against the original term submitted by the user.

HTTP POST Request

```
POST https://<Removed>/xrelevance-api-rest/relevance/disambiguate/expert/567
"apiKey:1db5b328-71a1-41e5-a5ee-b9cfc72c2999"
"Content-Type: application/json"
"Accept: application/json"

'["blood","skin"]'
```

JSON Response

```
Access-Control-Allow-Credentials :true
Access-Control-Allow-Origin :chrome-extension://fdmmgilgnpjigdojojpjoooidkmcomcm
Cache-Control :private
Content-Type :application/json;charset=UTF-8
Date :Thu, 13 Dec 2012 13:08:45 GMT
Expires :Thu, 13 Dec 2012 14:08:45 GMT
Last-Modified :Thu, 13 Dec 2012 13:08:45 GMT
Server :Apache-Coyote/1.1
Transfer-Encoding :chunked

{
    "queryId": null,
    "disambiguationTerms": [
        {
            "term": "vascular diseases",
            "score": 0.5
        },
        {
            "term": "contraceptive agents female",
            "score": 0.2
        }
    ]
}

0
```

Error Codes and Messages

Code	Name	Message
0	UnhandledException	Provides notification of uncaught exceptions.
2	IllegalCharacters	One of the arguments contains illegal characters.
4	EmptyArgument	Argument cannot be empty.
5	NullOrEmptyDisambiguateTerms	No terms were received for disambiguation.
8	LowLimit	Limit must be greater than 0.
9	LowOffset	Offset must be greater than or equal to 0.
10	MissingLimitParameter	Term limit required.
11	MissingOffsetParameter	Term offset required.
	OffsetExceededresults	Term offset exceeded the number of results.
29	ParameterTypeMismatch	Parameter is not the correct type.
39	RightsViolation	You are not authorized.
52	InvalidExpertId	Null or empty expertId.
60	NotAuthorizedException	Not authorized on a resource.
81	NullOrEmptyUrl	Null or empty url.
89	MethodDoesNotExist	Invalid method name.
91	CopntentNotOnline	Content corpus is not online.
110	MaxLimitExceeded	Limit must be lower than: x
404	NotFound	Resource not found.

getRelevantConcepts(POST)

Filters related domain expert terms and returns a list of relevant concepts.

Base URL

https://<Removed>/xrelevance-api-rest/relevance

Resource URI

/relevantConcepts/content/{contentId}

Path Variable	Description
`String contentId`	Index name. Identifies the target CRN for this method.

Request Parameter	Description
`String contentQuery`	Identifies the query for which we return a list of relevantTerm. Required. The value "*:*" can be used for the contentQuery field, if no value is available.
`String[] categoryFilter`	Identifies a list of categories used to filter the resulted terms.
`Integer docsLimit`	Specifies the maximum number of documents to return. Default = 10.

Request Body	Description
`ExpertDisambiguationModel query`	Contains terms to disambiguate as well as term limit and term offset parameters.

Returns

Returns an array of terms with relevancy scores.

HTTP POST Request

```
POST https://services3.xpatterns.com/xrelevance-api-rest/relevance/relevantConcepts/
content/
testContent?categoryfFilter=1:11&categoryfFilter=1:12&contentfFilter=age_i:[* TO
20]&contentQuery=male&offset=0&limit=5

"apiKey:b9cfc72c2999-41e5-a5ee-b9cfc72c2999"1db5b328-71a1
"Content-Type: application/json"
"Accept: application/json"

'["blood","skin"]'
```

JSON Response

```
Access-Control-Allow-Credentials :true
Access-Control-Allow-Origin :chrome-extension://fdmmgilgnpjigdojojpjoooidkmcomcm
Cache-Control :private
Content-Type :application/json;charset=UTF-8
Date :Thu, 13 Dec 2012 13:07:53 GMT
Expires :Thu, 13 Dec 2012 14:07:53 GMT
Last-Modified :Thu, 13 Dec 2012 13:07:53 GMT
```

```
Server :Apache-Coyote/1.1
Transfer-Encoding :chunked

{
    "queryId": null,
    "disambiguationTermsterms": [
        {
            "term": "contraceptive agents female",
            "score": 0.2,
            "annotatorCategoryPairs": [
                {
                    "category": "1:11",
                    "externalId": "externalId1"
                },
                {
                    "category": "1:13",
                    "externalId": "externalId2"
                }
            ]
        },
        {
            "term": "vascular diseases",
            "score": 0.5,
            "annotatorCategoryPairs": [
                {
                    "category": "2:23",
                    "externalId": "externalId3"
                },
                {
                    "category": "2:21",
                    "externalId": "externalId4"
                }
            ]
        }
    ]
}
```

Error Codes and Messages

Code	Name	Message
0	UnhandledException	Provides notification of uncaught exceptions.
8	LowLimit	Limit must be greater than 0.
9	LowOffset	Offset must be greater than or equal to 0.
10	MissingLimitParameter	Term limit required.
11	MissingOffsetParameter	Term offset required.
19	OffsetExceededresults	Term offset exceeded the number of results.
28	NotExistingDocumentId	Document not found.
29	ParameterTypeMismatch	Parameter is not the correct type.
39	RightsViolation	You are not authorized
54	InvalidContentId	Null or empty content id.

Code	Name	Message
55	MissingContent	Missing content id parameter.
59	InvalidRight	Null or empty right name.
60	NotAuthorizedException	Not authorized on a resource.
81	NullOrEmptyUrl	Null or empty url.
89	MethodDoesNotExist	Invalid method name
91	ContentNotOnline	Content corpus not online yet.
404	NotFound	No Found.

searchContent

This method returns relevant content that can be filtered, sorted, and presented in a multitude of ways based on the values of input parameters.

Base URL

https://<Removed>/xrelevance-api-rest/relevance

Resource URI

/searchContent/content/{contentId}

Path Variable	Description
String contentId	Identifies the target for this query.

Request Parameter	Description
String query	Identifies a specific query. Required.
String [] filters	Custom fields used in context.
String [] facets	Category into which to group relevant content (e.g., Manufacturer). Supports faceted browsing and enables users to refine search results.
String [] facetFields	Constraints or facet values that pertain to the specified facet (e.g., Ford, Toyota, BMW).
Integer offset	Specifies a zero-based offset into the relevant content at which to begin retrieving (e.g., for pagination). Default = 0.
Integer limit	An optional maximum number of relevant content items to return as a result of the query starting at offset. Default = 10.
String [] sort	Specifies the order in which to sort results. Provide the sort field followed by a space and the sort method. Default = score descending.
String [] fields	String array that contains one or more partner-defined fields.
String facetDateStart	Start date of faceted data.
String facetDateEnd	End date of faceted data.
Integer facetDateGap	The size of each date range expressed as an interval to be added to the lower bound of the facet date.
String [] categoryFilter	The unique Id of plugin and category ex:1:3 (plugin1 and the 3rd category)
Boolean spellCheckerEnabled	Toggles spell checker.on or off. Default = False.
Boolean showCategories	Default = False.

Boolean Queries

Query	xRelevance Query
`Term1 AND Term2 AND Term3`	query=+Term1 +Term2 +Term3
`Term1 OR Term2 OR Term3`	query=Term1 Term2 Term3
`"Multi word ngram 1" AND "Multi word ngram 2"`	query=+"Multi word ngram 1" +"Multi word ngram 2"
`"Multi word ngram 1" OR "Multi word ngram 2"`	query="Multi word ngram 1" "Multi word ngram 2"
`NOT Term1`	query=-Term1

Returns

Returns relevant content.

HTTP GET Request

```
GET https://<removed>/xrelevance-api-rest/relevance/searchContent/content/
fullWiki?query=mock
query&offset=0&limit=10&facet=age_i:23&facet=age_i:30&facetField=age_i&field=title&c
ategoryFilter=1:3&filter=age_i:[1 TO
*]&showCategories=true&spellCheckerEnabled=true&sort=score%20desc

Content-Type:application/json; charset=utf-8
X-Request:JSON
dataType:json
apikey:5ba70490-8744-4912-a8bf-5ad9bf79969d
```

JSON Response

```
Content-Type :application/json;charset=UTF-8
Date :Thu, 13 Dec 2012 13:37:50 GMT
Server :Apache-Coyote/1.1
Transfer-Encoding :chunked

{
    "query": "mock query",
    "content": [
        {
            "categories": [
                "1:11",
                "1:12",
                "2:21"
            ],
            "fields": {
                "title": "1st title"
            },
            "documentId": "id1",
            "score": 4.05
        },
        {
            "categories": [
                "2:21",
                "2:22",
                "1:14"
            ],
            "fields": {
                "title": "second title"
            },
            "documentId": "id2",
            "score": 3.45
        },
        {
            "categories": null,
            "fields": {
                "title": "3rd title"
            },
            "documentId": "id3",
            "score": 3.25
        }
    ],
    "queryFacets": {
        "age_i:23": 5,
        "age_i:30": 4
    },
    "fieldFacets": {
        "age_i": {
            "age_i:14": 2,
            "age_i:23": 5,
            "age_i:30": 4
        }
    },
    "spellCheckResponse": {
        "correctlySpelled": false,
        "spellcheckSuggestions": [
            {
                "collationQueryString": "mocking query",
                "numberOfHits": 11
            },
            {
                "collationQueryString": "mocked query",
                "numberOfHits": 21
            },
            {
                "collationQueryString": "mock queries",
                "numberOfHits": 6
            }
        ]
    },
    "queryId": null,
    "totalFound": 3
}
```

Error Codes and Messages

Code	Name	Message
0	UnhandledException	Provides notification of uncaught exceptions.
29	ParameterTypeMismatch	Parameter is not the correct type.
54	InvalidContentId	Content id does not exist.
55	MissingContent	Missing content id parameter.
59	InvalidRight	Null or empty right name.
60	NotAuthorizedException	HTTPS URLs not supported.
81	NullOrEmptyUrl	Null or empty url string detected.

Code	Name	Message
83	NoContentRetrievedFromUrl	No textual content extracted from URL
84	ErrorRetrievingOrProcessingContentFromUrl	Error retrieving or processing content from url.
91	ContentNotOnline	Content corpus not online yet.
404	NotFound	Not Found.

publishNote

Notifies the system that coding is complete.

Base URL

https://<Removed>/

Resource URI

note/{emrId}/publish

Path Variable	Description
int emrId	Identifies the note to publish.

Returns

Returns the noteId

HTTP POST Request

```
PUT https://<Removed>/cac-api-rest/note/4752/publish HTTP/1.1
Accept: application/json
apikey: 29812723-98b9-47f5-bdb3-d872ee76bfec
User-Agent: Java/1.6.0_33
Host: <Removed>
Connection: keep-alive
```

JSON Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 15 Feb 2013 20:06:24 GMT

d
{"id":"4752"}
0
```

Error Codes and Messages

This table contains a list of the most common error codes and HTTP response codes returned by xPatterns for this method.

Code	Name
0	**UnhandledException** This exception likely caught us off guard.
1	**ParameterTypeMismatch** Parameters in the request do not match the data type that's expected. For example, this error happens if requests contain integers when Strings are expected.
2	**InvalidBodyRequestParameter** Parameters are likely missing, spelled incorrectly, or appear out of order in the request.

Code	Name
4	**InvalidCredentials** We are unable to find a match in our system for the username and password combination you provided.
6	**NotAuthorizedException** Be sure you're using the correct API key for the method you're calling. Admin features require a different access level than coder features.
7	**NullOrEmptyUrl** The url in the request seems to be missing.
8	**InvalidTag** Please check the entity names (e.g., encounters, notes) included in your request.
400	**Bad Request** The request could not be understood by the server due to malformed syntax (such as a bad query string).
401	**Unauthorized** A user token has expired, is invalid, or attempted to access an endpoint to which it does not have access.
404	**NotFound** The resource requested was not found or the URL used to make the call was invalid.
500	**Internal Server Error** This means something went wrong on our end. Please try again or contact your Atigeo Customer Service Representative.
503	**Service Unavailable** We're experiencing temporary down-time.

Java Sample Code

publishNote

listEncounters

Lists encounters in the system and provides parameters that support filtering and sorting.

Base URL

https://<Removed>/

Resource URI

encounter/list

Request Parameters	Description
Integer offset	Specifies an optional offset into results, at which to begin retrieval (e.g., for pagination). Required.
Integer limit	Restricts the maximum number of encounters to return starting at the specified offset. Required.
String status	Specifies the status of encounters that will be returned. Values include assigned, unassigned, complete, published, and rejected. Optional.
String period	Specifies a date range.
String startDate	Specifies first date of the date range for which you are filtering the encounters.
String endDate	Specifies last date of the date range for which you are filtering the encounters.
String sortField	Specifies the field by which to sort the encounters.
Boolean ascending	Specifies the sorting direction for the returned data. Sorting direction defaults to descending and can be changed to ascending by setting this parameter to *true*.
String tag	Friendly name for an encounter.
String productType	Specifies the product type of encounters that will be returned. Values include outpatient, inpatient, and physician. defaultValue = “outpatient”. Optional.

Returns

A list of encounters and associated metadata, including label, importUserId, assignedUserId, assignedUserName, emrId, lastModified, status, tags, and a codeable flag.

HTTP GET Request

```
GET https://<Removed>/cac-api-rest/encounter/
list?offset=0&limit=1000&productType=outpatient HTTP/1.1
Accept: application/json
apikey: 3a52b9e7-42f3-4d9d-8f55-185c4b92a0f5
User-Agent: Java/1.6.0_33
Host: <Removed>
Connection: keep-alive
```

JSON Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 12 Feb 2013 05:04:09 GMT

2000
```

```
{"encounterLabels":[{"label":"11-
1360640725418Samples.zip","importUserId":11,"assignedUserId":11,"assignedUserName":"
defaultUser","encounterId":3660,"createdDate":1360640727579,"status":"assigned","adm
issionDate":1360381527579,"dischargeDate":1360640727579,"reimbursement":null,"specia
lty":null},{"label":"Test_41158b5","importUserId":11,"assignedUserId":11,"assignedUs
erName":"defaultUser","encounterId":3658,"createdDate":1360639694936,"status":"assig
ned","admissionDate":1360380494936,"dischargeDate":1360639694936,"reimbursement":nul
l,"specialty":null},{"label":"Test_2b09eef","importUserId":11,"assignedUserId":11,"a
ssignedUserName":"defaultUser","encounterId":3654,"createdDate":1360639532238,"statu
s":"assigned","admissionDate":1360380332238,"dischargeDate":1360639532238,"reimburse
ment":null,"specialty":null},{"label":"Test_3448078","importUserId":11,"assignedUser
Id":11,"assignedUserName":"defaultUser","encounterId":3653,"createdDate":13606395136
03,"status":"complete","admissionDate":1360380313603,"dischargeDate":1360639513603,"
reimbursement":null,"specialty":null},{"label":"Test_13b57aa","importUserId":11,"ass
ignedUserId":11,"assignedUserName":"defaultUser","encounterId":3651,"createdDate":13
60639500143,"status":"rejected","admissionDate":1360380300143,"dischargeDate":136063
9500143,"reimbursement":null,"specialty":null},{"label":"Test_d7c37ba","importUserId
":11,"assignedUserId":11,"assignedUserName":"defaultUser","encounterId":3649,"create
dDate":1360639467346,"status":"rejected","admissionDate":1360380267346,"dischargeDat
e":1360639467346,"reimbursement":null,"specialty":null},{"label":"11-
1360638258840CASE 9.zip-
1358895477016Samples.zip","importUserId":11,"assignedUserId":11,"assignedUserName":"
defaultUser","encounterId":104854,"createdDate":1358895481199,"status":"assigned","a
dmissionDate":1358636281199,"dischargeDate":1358895481199,"reimbursement":null,"spec
ialty":null}],"count":332}
0
```

Error Codes and Messages

This table contains a list of the most common error codes and HTTP response codes returned by xPatterns for this method.

Code	Name
0	**UnhandledException** This exception likely caught us off guard.
1	**ParameterTypeMismatch** Parameters in the request do not match the data type that's expected. For example, this error happens if requests contain integers when Strings are expected.
2	**InvalidBodyRequestParameter** Parameters are likely missing, spelled incorrectly, or appear out of order in the request.
4	**InvalidCredentials** We are unable to find a match in our system for the username and password combination you provided.

Code	Name
6	**NotAuthorizedException** Be sure you're using the correct API key for the method you're calling. Admin features require a different access level than coder features.
7	**NullOrEmptyUrl** The url in the request seems to be missing.
8	**InvalidTag** Please check the entity names (e.g., encounters, notes) included in your request.
400	**Bad Request** The request could not be understood by the server due to malformed syntax (such as a bad query string).
401	**Unauthorized** A user token has expired, is invalid, or attempted to access an endpoint to which it does not have access.
404	**NotFound** The resource requested was not found or the URL used to make the call was invalid.
500	**Internal Server Error** This means something went wrong on our end. Please try again or contact your Atigeo Customer Service Representative.
503	**Service Unavailable** We're experiencingtemporary down-time.

Java Sample Code

listEncounters

addNoteContent

Adds a new note to an existing encounter.

Base URL

https://<Removed>/

Resource URI

encounter/{encounterId}/notetext

Path Variables	Description
int encounterId	Identifies a specific encounter.

Request Parameters	Description
String productType	Specifies the product type of the encounter containing the notes that will be returned. Values include outpatient, inpatient, and physician. defaultValue = “outpatient”. Optional.

Request Body	Description
NoteContent note	Contains text that will be added with the new note to an existing encounter.

Returns

Returns the noteId of the newly added note.

HTTP POST Request

```
POST https://<Removed>/cac-api-rest/encounter/3661/notetext?productType=outpatient
HTTP/1.1
Accept: application/json
apikey: 29812723-98b9-47f5-bdb3-d872ee76bfec
Content-Type: application/xml
User-Agent: Java/1.6.0_33
Host: <Removed>
Connection: keep-alive
Content-Length: 107

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><noteContent><text>Heart
Attack</text></noteContent>
```

JSON Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 12 Feb 2013 18:30:51 GMT

d
```

```
{"id":"4705"}
0
```

Error Codes and Messages

This table contains a list of the most common error codes and HTTP response codes returned by xPatterns for this method.

Code	Name
0	**UnhandledException** This exception likely caught us off guard.
1	**ParameterTypeMismatch** Parameters in the request do not match the data type that's expected. For example, this error happens if requests contain integers when Strings are expected.
2	**InvalidBodyRequestParameter** Parameters are likely missing, spelled incorrectly, or appear out of order in the request.
4	**InvalidCredentials** We are unable to find a match in our system for the username and password combination you provided.
6	**NotAuthorizedException** Be sure you're using the correct API key for the method you're calling. Admin features require a different access level than coder features.
7	**NullOrEmptyUrl** The url in the request seems to be missing.
8	**InvalidTag** Please check the entity names (e.g., encounters, notes) included in your request.
400	**Bad Request** The request could not be understood by the server due to malformed syntax (such as a bad query string).
401	**Unauthorized** A user token has expired, is invalid, or attempted to access an endpoint to which it does not have access.
404	**NotFound** The resource requested was not found or the URL used to make the call was invalid.
500	**Internal Server Error** This means something went wrong on our end. Please try again or contact your Atigeo Customer Service Representative.
503	**Service Unavailable** We're experiencing temporary down-time.

Java Sample Code

addNoteContent

IBS Agreement Management Web Service API Reference Guide

The Agreement Management Web Service exposes methods for working with IBS configuration data as well as the business data associated with *agreements* and *products*. also known as agreement details. These interfaces give your company complete control over your configuration and how you handle your business data. They also enable you to determine how the your system exchanges information with external systems and other applications. Through the API, it's possible to replicate all IBS actions available from the CSR and Configuration modules.

- IAgreementManagementConfigurationService Interface—provides methods for creating, retrieving, and updating IBS configuration data for your agreements and products.
- IAgreementManagementService Interface—provides methods for: creating, retrieving, and updating Agreements and AgreementDetails, linking hardware and software to an agreement, changing the status of an agreement per configured event and reason, scheduling updates, and for processing pre-payment quotes.

See Also

- The WCF Service Model Endpoints—describes the service endpoints (address, binding, and contract) established for communication with the IBS Web services.
- Namespaces—describes how interfaces and business objects are organized in IBS..
- Lookup Lists—describes the mechanism for retrieving data from the configuration module.
- ServiceFactory—describes the ServiceFactory, which returns WCF service proxies.
- AuthenticationHeader—describes the IBS login process.
- GetServiceLocation—describes the best practice for retrieving fully qualifies URIs of the Web services.

IAgreementManagementConfigurationService

The IAgreementManagementConfigurationService interface provides methods for creating, retrieving, and updating setup details for Agreements, such as:

- AgreementDetailSpearEvent, AgreementDetailSpearRuleGroup, and AgreementSpear, which are the configured values that determine what happens to the agreement when events are triggered or update reasons are specified.
- AgreementSequences, the configured flowchart for processing agreement entities: the shipping orders, work orders, products, and other entities that IBS automatically generates when required by a particular agreement. The flowchart defines conditions and corresponding updates to make to each of these entities at each possible step or junction in the overall process. Usually, some updates require manual input, whereas others are performed automatically.

- AgreementStatus, AgreementType, ContractPeriod, RenewalCondition, RenewalConfiguration, and UnitOfMeasurement, which are the configured values used when creating Agreements.

I[Entity]ConfigurationService works with configuration data. For example, ICustomersConfigurationService provides methods such as CreateValidAddress(), GetKeywordTypes(), and GetProvinceByCountry().

Each ConfigurationService also has a GetLookups() method which takes one LookupList value as a parameter and returns the values in the Configuration module for that identifier. For example, calling GetLookups(LookupLists.Country) will return the collection of countries stored in Configuration. Invoking GetLookups(LookupLists.ReasonAddressUpdate) will return the collection of valid reasons fLookupLists - configuration data identifiers used by business entities

The ConfigurationService has a GetLookups(LookupLists lookupList) method which returns the data associated with the identifier.or updating an address.

- **endpoint name**="ICustomersService"
- **address**=http://ibsinterprit/ASM/ALL/Customers.svc
- **binding**="basicHttpBinding"
- **contract**="PayMedia.ApplicationServices.Customers.ServiceContracts.ICustomersServ ice”

  <Removed Documentation>

IAgreementManagementService Interface

The IAgreementManagementService interface provides methods for creating, retrieving, and updating, Agreements, AgreementDetails, including linking associated hardware and software to the Agreement, changing the status of an agreement per a configured event and reason, scheduling updates to occur at a future date or time, and processing pre-payment quotes.

To completely add a new active customer, an Agreement must be be created and associated with the Customer (per the Agreement.CustomerId property). The Agreement must have an AgreementDetail. Products are added to the AgreementDetail by adding a ComericalProduct, which can have multiple TechnicalProducts. Specific software, devices, and a la carte products are linked to the TechnicalProducts by setting the TechnicalProductId of the SoftwarePerAgreementDetail, DevicePerAgreementDetail, or ALaCarteProduct to the appropriate TechnicalProduct.Id value.

Capturing a Product

The easiest and most common way to capture a product is to use the IAgreementManagementService.CreateAgreementDetail method. The CreateAgreementDetail method takes an AgreementDetail object and reason, and performs the create action.

However, when creating a user interface that allows price quoting, the IAgreementManagementService.ManageProductCapture product may be easier, as it takes the same data object as the IAgreementManagementService.GetManageFullProductDetail method. The ManageProductCapture method takes more parameters than the CreateAgreementDetail method, but can perform a product capture, reconnect, cancel, upgrade, or downgrade, all based on the customer's original products versus the data included in the parameters.

Both the CreateAgreementDetail and ManageProductCapture methods automatically generate DevicePerAgreementDetails to link any hardware for the product to a physical device and/or software. Both methods also attempt to generate SoftwarePerAgreementDetails, ALaCarteProducts (if required), and an AgreementSequenceInstance object that is responsible for handling the status changes of any products, shipping orders, or work orders that were created in the process. While the CreateAgreementDetail method does not support the creation of shipping and work orders, the ManageProductCapture method does create shipping and work orders, if either are required.

While both of these methods were designed to use a sandbox as a temporary workspace, the CreateAgreementDetail method may be called without a sandbox. However, if used without a sandbox, the CreateAgreementDetail method does not generate any shipping or work orders. The ManageProductCapture method must always be used with a sandbox.

Reconnecting a Product

Like capturing a product, reconnecting a product can be done multiple ways. The easiest and most common way to reconnect a product is to use the IAgreementManagementService.ReconnectAgreementDetails method, which simply requires a list of Ids for the AgreementDetails to reconnect. The IAgreementManagementService.ManageProductCapture method can also be used to reconnect a product, but requires more data.

For more information on the ReconnectAgreementDetails and ManageProductCapture methods, see Chapter 6: AgreementManagement Service on page 43.

For more information on the CreateAgreementDetail and ManageProductCapture methods, see Chapter 6: AgreementManagement Service on page 43.

See Also For the methods to create CommercialProducts and TechnicalProducts, see...<removed links>

Retrieving a List of Products

To retrieve a list of all products purchased by a Customer, use the GetAgreementDetailsForCustomer method.

See Also

- To retrieve a list of all CommercialProducts from the Configuration Module, use the IProductCatalogConfigurationService.GetCommercialProducts method.
- To retrieve a list of all TechnicalProducts from the Configuration Module, use the IProductCatalogConfigurationService.GetTechnicalProducts method.

ActivateProduct

Activates the product associated with the provided agreementDetailId, and applies the SPEAR associated with Activation and the provided reason. Returns the updated AgreementDetail.

```
AgreementDetail ActivateProduct
{
int agreementDetailId,
int reason
};
```

Related Methods

- ActivateProducts()

ActivateProducts

Activates the products associated with the provided agreementDetailIds, and applies the SPEAR associated with Activation and the provided reason. Returns the updated AgreementDetailCollection.

```
AgreementDetailCollection ActivateProducts
{
int[] agreementDetailIds,
int reason
};
```

Related Methods

- ActivateProduct()
-

DisconnectAgreementDetails

Changes the status of the provided AgreementDetails to Disconnect. To change the status to Disconnect at a future date or time, use the ScheduleDisconnectAgreementDetails Method.

```
AgreementDetailCollection DisconnectAgreementDetails
{
int[] agreementDetailIds,
int reason
};
```

Related Methods

- CancelAgreementDetails()
- CreateAgreementDetails()

- GetAgreementDetail()
- GetAgreementDetails()
- ReconnectAgreementDetails()
- ScheduleDisconnectAgreementDetails()
- ScheduleReconnectAgreementDetails()
- SleepAgreementDetails()
- UpdateAgreementDetails()

GetAllDeviceEntitlements

Returns the DeviceEntitlements associated with the provided agreementDetailId or software on other AgreementDetails that are running on hardware of the provided agreementDetailId.

If true, returns DeviceEntitlements that are linked with the provided agreementDetailId that are associated with other AgreementDetails whose Status.CanWatch is true. If false, returns DeviceEntitlements that are linked with the provided agreementDetailId that are associated with other AgreementDetails whose Status.CanWatch is false.

```
DeviceEntitlementCollection GetAllDeviceEntitlements
{
int agreementDetailId,
bool isActive
};
```

Related Methods

- GetDeviceEntitlements()

GetSoftwareForAgreementDetails

Returns the SoftwarePerAgreementDetails associated with the provided agreementDetailIds.

To return the SoftwarePerAgreementDetails for a single AgreementDetailId, use the GetSoftwarePerAgreementDetailForAgreementDetail Method.

```
SoftwarePerAgreementDetailCollection GetSoftwareForAgreementDetails
{
int[] agreementDetailIds,
int page
};
```

Related Methods

- GetSoftwareForAgreementDetailByAD_IdAndTechProdId()
- GetSoftwareForAgreementDetailById()
- GetSoftwareForAgreementDetailByIds()
- GetSoftwareForAgreementDetailsByDpADId()

- GetSoftwarePerAgreementDetailByLoSId()
- GetSoftwarePerAgreementDetailHistory()
- GetSoftwarePerAgreementDetailHistoryDetails()
- MusicNet CAM API Reference Guide()

ManageProductCapture

Used primarily by the IBS v6 Manage Product screen, this method handles the creation of offers and agreement details. While the recommended method for capturing a product is to use the IAgreementManagementService.CreateAgreementDetail method, the ManageProductCapture may be preferred when creating a user interface that allows price quoting, as as it takes the same data object as the IAgreementManagementService.GetManageFullProductDetail method. In addition to capturing a product, the ManageProductCapture method can perform a product capture, reconnect, cancel, upgrade, or downgrade, all based on the customer's original products versus the data included in the provided detailParams.

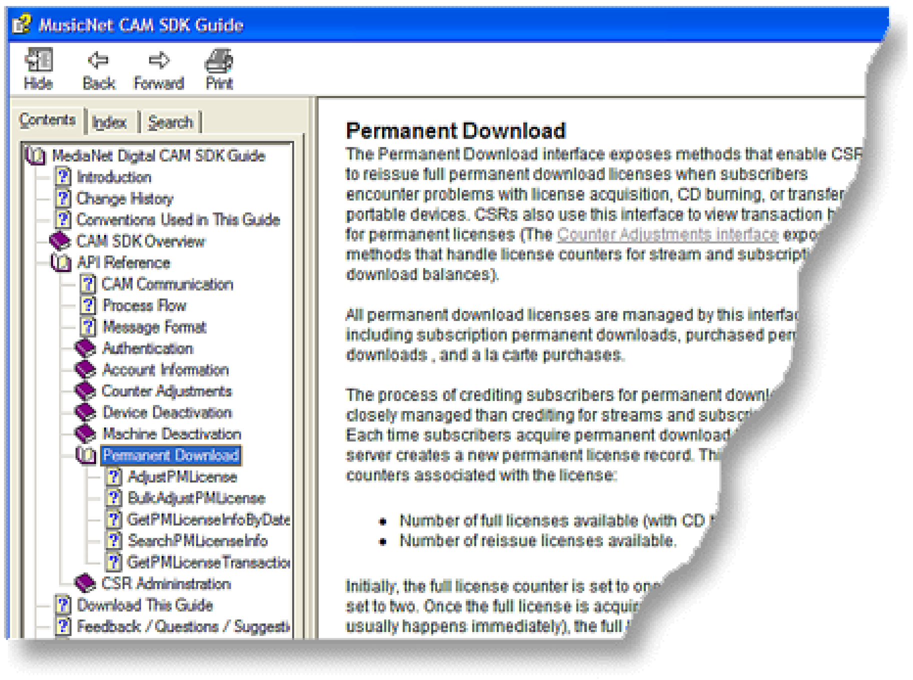
Both the CreateAgreementDetail and ManageProductCapture methods automatically generate DevicePerAgreementDetails to link any hardware for the product to a physical device and/or software. Both methods also attempt to generate SoftwarePerAgreementDetails, ALaCarteProducts (if required), and an AgreementSequenceInstance object that is responsible for handling the status changes of any products, shipping orders, or work orders that were created in the process. While the CreateAgreementDetail method does not support the creation of shipping and work orders, the ManageProductCapture method does create shipping and work orders, if either are required.

While both of these methods were designed to use a sandbox as a temporary workspace, the CreateAgreementDetail method may be called without a sandbox. However, if used without a sandbox, the CreateAgreementDetail method does not generate any shipping or work orders. The ManageProductCapture method must always be used with a sandbox.

For more information on capturing a product, see Capturing a Product on page 40. For more information on the CreateAgreementDetail method, see CreateAgreementDetail.

```
ProductCaptureResults ManageProductCapture
{
ProductCaptureParams detailParams
};
```

MusicNet CAM API Reference Guide



MusicNet CAM SDK Guide

- MediaNet Digital CAM SDK Guide
  - Introduction
  - Change History
  - Conventions Used in This Guide
  - CAM SDK Overview
  - API Reference
    - CAM Communication
    - Process Flow
    - Message Format
    - Authentication
    - Account Information
    - Counter Adjustments
    - Device Deactivation
    - Machine Deactivation
    - Permanent Download
      - AdjustPMLicense
      - BulkAdjustPMLicense
      - GetPMLicenseInfoByDate
      - SearchPMLicenseInfo
      - GetPMLicenseTransactio
    - CSR Administration
  - Download This Guide
  - Feedback / Questions / Suggesti

Permanent Download

The Permanent Download interface exposes methods that enable CSR
to reissue full permanent download licenses when subscribers
encounter problems with license acquisition, CD burning, or transfer
portable devices. CSRs also use this interface to view transaction h
for permanent licenses (The Counter Adjustments interface expo
methods that handle license counters for stream and subscripti
download balances).

All permanent download licenses are managed by this interfa
including subscription permanent downloads, purchased per
downloads , and a la carte purchases.

The process of crediting subscribers for permanent downl
closely managed than crediting for streams and subscri
Each time subscribers acquire permanent download
server creates a new permanent license record. Thi
counters associated with the license:

- Number of full licenses available (with CD b
- Number of reissue licenses available.

Initially, the full license counter is set to on
set to two. Once the full license is acquir
usually happens immediately), the full l

Permanent Download

The Permanent Download interface exposes methods that enable CSRs to reissue full permanent download licenses when subscribers encounter problems with license acquisition, CD burning, or transfer to portable devices. CSRs also use this interface to view transaction history for permanent licenses (The Counter Adjustments interface exposes methods that handle license counters for stream and subscription download balances).

All permanent download licenses are managed by this interface, including subscription permanent downloads, purchased permanent downloads , and a la carte purchases.

The process of crediting subscribers for permanent downloads is more closely managed than crediting for streams and subscription downloads. Each time subscribers acquire permanent download licenses, the CAM server creates a new permanent license record. This record contains two counters associated with the license:

- Number of full licenses available (with CD burn/transfer rights).
- Number of reissue licenses available.

Initially, the full license counter is set to one, and the reissue counter is set to two. Once the full license is acquired by a client application (which usually happens immediately), the full license counter is automatically decremented to zero.

If the reissue counter is set to zero, we return an error.

✓ **Note** In the future, the initial number of full licenses and reissues may change.

Reissues are based on component ids and work with tracks and albums regardless of how they're purchased.

When customers report problems with permanent downloads, this interface manages the process of decrementing the reissue counter by one and incrementing the full license counter by one. If the reissue counter is at zero, subscribers must either use another permanent download from their unused balance or purchase the download a la carte.

Permanent download license information is organized by track and includes track-related metadata, such as artist, song title, and album information.

Table 1: Permanent Download Methods

PermanentDownload Methods	Description
AdjustPMLicense	Reissues a permanent download license with full rights.
BulkAdjustPMLicense	Reissues permanent download licenses with full rights for an array of tracks.
GetPMLicenseInfoByDate	Retrieves a list of permanent licenses issued within a specified range of dates.
SearchPMLicenseInfo	Searches song titles and artist names for a subscriber's permanent download license information.
GetPMLicenseTransactionHistory	Retrieves permanent download license adjustment history for a subscriber's current and/or previous subscription period.

AdjustPMLicense

Adjusts the reissue counter for a specific track to enable a subscriber to get a new license for a permanent download. It decrements the reissue counter by one and increments the full license counter by one.

Reissues are based on component ids and work with tracks and albums regardless of how they're purchased.

Syntax

AdjustPMLicense(

string *sessionID*,

string *retailerCustomerId*,

string *trackId*,

string *notes*,

Parameters

sessionID [in]

Current session ID.

retailerCustomerId [in]

Unique identifier for customer requesting counter adjustments.

trackId [in]

Track identifier.

notes [in]

Comments from the CSR.

Remarks

This method returns a boolean value for success (true) or failure (false).

Example Request

AdjustPMLicense requests are sent in this format:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:types="http://172.16.11.31/jboss-net/services/CamService/encodedTypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<q1:ActivateUser xmlns:q1="http://soap.camsdk.musicnet.com">
		<sessionId xsi:type="xsd:string">719bcd5d6</sessionId>
		<camUser xsi:type="xsd:string">sample-csr</camUser>
</q1:ActivateUser>
</soap:Body>
</soap:Envelope>
Example Response
AdjustPMLicense responses arrive in this format:
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
		 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
		xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
    <ns1:ActivateUserResponse xmlns:ns1="urn::camsdk::Admin"
    SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
       <return xsi:type="xsd:boolean">true</return>
    </ns1:ActivateUserResponse>
		</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

BulkAdjustPMLicense

Adjusts permanent download licenses with full rights. This method adjusts a subscriber’s permanent download full licenses and reissue counters for each track in an array; it decrements the reissue counters by one and increments the full license counters by one.

Reissues are based on component ids and work with tracks and albums regardless of how they're purchased.

Syntax

BulkAdjustPMLicense(

string *sessionID,*

string *retailerCustomerId,*

string [ ] *trackId,*

string *notes,*

string *response*)

Parameters

sessionID [in]

Current session ID.

retailerCustomerId [in]

Unique identifier for customer requesting counter adjustments.

trackId [in]

Track identifier. The BulkAdjustment method passes an array of track IDs.

notes [in]

Comments from the CSR.

response [out]

Response to the method call. We return the track IDs and a Boolean value (which indicates success or failure) for each track in the array.

Constraint

The track ID array can store up to 1000 track IDs.

Example Request

BulkAdjustPMLicense requests are sent in this format:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://172.16.11.31/jboss-net/services/CamService"
xmlns:types="http://172.16.11.31/jboss-net/services/CamService/encodedTypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<q1:BulkAdjustPMLicense xmlns:q1="http://soap.camsdk.musicnet.com">
<sessionId xsi:type="xsd:string">719bcd5def2af4e6</sessionId>
<retailerCustomerId xsi:type="xsd:string">apptest000000120040213164051.242</retailerCustomerId>
<trackIds href="#id1"/>
<notes xsi:type="xsd:string">Sample Adjust PM License</notes>
</q1:BulkAdjustPMLicense>
<soapenc:Array id="id1" soapenc:arrayType="xsd:string[1]">
<Item>32004</Item>
</soapenc:Array>
```

```
</soap:Body>
</soap:Envelope>
```

Example Response

BulkAdjustPMLicense responses arrive in this format:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
        <ns1:BulkAdjustPMLicenseResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://soap.camsdk.musicnet.com">
                  <ns1:BulkAdjustPMLicenseReturn xsi:type="xsd:string">
<Response>
        <Component>
                <Component-Id>1234</Component-Id>
                <Adjustment-Succeeded>TRUE</Adjustment-Succeeded>
        </Component>
        <Component>
                <Component-Id>23456</Component-Id>
                <Adjustment-Succeeded>FALSE</Adjustment-Succeeded>
        </Component>
        <Component>
                <Component-Id>33002</Component-Id>
                <Adjustment-Succeeded>FALSE</Adjustment-Succeeded>
        </Component>
</Response>
</ns1:BulkAdjustPMLicenseReturn>
        </ns1:BulkAdjustPMLicenseResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Integration Guides

Some of the writing samples in this section are rough approximations of the actual documentation that I published. Several of my samples are publicly available, but in some cases, I had to remove illustrations, product names, company names, tables, and entire sections, so the copyright owner would grant me permission to use excerpts in my portfolio.

Authentication Integration

This chapter discusses two authentication topics:

a Authentication with Encrypted Tokens
Describes how authentication tokens can be used by a third party music service to authenticate with the MusicNet platform.

b Distributed Authentication
Describes the authentication reconciliation process used by MusicNet.

Authentication with Encrypted Tokens

Certain MusicNet service APIs accept method calls from only authenticated clients. MusicNet must verify that the authentication token a client presents to the MusicNet service in the login API is valid. Only trusted authentication providers or token issuer services (hosted by the partner) can verify user credentials (i.e., user name, password) and issue tokens. The data integrity of the token must be guaranteed, and if needed, data confidentiality must also be guaranteed; to guarantee both, we use encrypted authentication tokens.

Encrypted Tokens

Encrypted authentication tokens are tamper resistant and their token payloads are secure. The token payload includes a random salt, a magic string , a unique user id, and a time stamp. The token issuer uses a symmetric key to encrypt this payload. Upon receiving the token, MusicNet decrypts the token, validates it, and extracts the user id. This solution assumes that the symmetric key (or pass phrase from which the symmetric key is derived) can be securely exchanged between the token issuer and MusicNet.

Encryption Parameters

The algorithm used is AES-CBC with 128 bit key and 128 bit block size. Consequently, the initialization vector is 16 bytes. The padding is PCKS#5.

.Net Framework Sample Code

The code sample below generates an authentication token. The inputs of the program are the base64-encoded 128-bit AES key and the userId:

.NET will use PCKS#7 – identical to PCKS#5 up to 256 bytes blocks.

Token Format

The plaintext token has the following format:

plaintext = salt | | (magic string) | | userId '|' timestamp

salt = 16 random bytes

magic string = "TKN"

timestamp = UTC date time in the format YYYY-MM-DD HH:MM:SSUTC

userId = unique alphanumeric user or customer id

| | = concatenation

The encrypted token has the following format:

token = Base64((init vector) || ciphertext)

init vector = 16 bytes

ciphertext = AES encryption of the plaintext

Token Validation

When a token arrives, the MusicNet application server:

c Decodes the base64 token and parses out the initialization vector and the cipher text.

d Decrypts the cipher text;

e Removes the salt;

f Compares the magic string with "TKN" (case sensitive comparison) to validate decryption of the token;

g Parses out the timestamp and the user id.

Key Management

The current implementation uses a simple key management solution. The AES keys are 128-bit long and base64 encoded. They must be exchanged with MusicNet using an out-of-band mechanism, which is not defined in this document.

The MusicNet application server can manage two keys at one time. The first key is the active key, which is tried first in the token decryption step. The second key is the accepted key that's tried if decryption with the first key fails.

This mechanism allows for changing the production keys, so old tokens are still authenticated for a period of time, until the new set of keys are propagated in the system.

More advanced key management mechanisms are TBD.

Distributed Authentication

This section describes the authentication reconciliation process for the MusicNet Service Platform. We discuss two solutions:

a Client sends opaque token, and MusicNet delegates authentication;

b Client sends a token, and MusicNet validates the token locally (Preferred).

Solution 2 (local validation of tokens) has several advantages and no significant disadvantages or weaknesses. The solution is scalable, and the effort to implement it is relatively low, depending on your architecture. We can provide reference code to create and validate tokens using Java and Perl.

Integration Points

You must address two points of integration to enable users to authenticate with the MusicNet Service Platform:

- **a** MusicNet E-commerce Management (Client sign-on/Ecommerce) Authentication and;
- **b** MusicNet CAM Authentication.

You can satisfy both integration points with a single sign-on solution.

Single Sign In

Each user should have a single set of credentials to log into the service, and this should be done only through the client-user interface.

Upon successful sign-in to your service, subsequent authentication to the MusicNet service is transparent to users. In this context, sign-in means the process by which users send their credentials (user names and passwords managed by your organization) to our server, which verifies those credentials.

Authentication in the MusicNet Service

MusicNet is not an authentication provider, and we do not store user names or passwords in our system; authentication services are the sole responsibility of your organization.

Rather than use profile information to authenticate users, MusicNet uses authentication tokens as user credentials. We expose a Login web-service method that accepts tokens as input parameters. These tokens are verified either locally or by calling a remote authentication provider, which will most likely be your organization.

Ideally, the authentication token is encrypted to provide data integrity, confidentiality if needed, and non-repudiation. Upon successful validation, this process must yield a unique userId (generated by your service), which is an alphanumeric value that uniquely identifies users within your company-user namespace.

After successful authentication, the service checks for a MusicNet subscription account, authorizes the user, creates a session with the MusicNet service, and returns a session key to the client. Since the session key must be protected against tampering and eavesdropping, login to MusicNet must use HTTPS as the transport protocol.

Authentication in the Partner Service

You'll expose your authentication schema using a Login function that takes the following parameters:

- username,
- password,
- sessionId,
- and playerId.

The login function validates these credentials with your authentication provider. Upon successful authentication, the userId is sent back to the client.

MusicNet Delegates Authentication

This first solution requires an opaque token that can be passed to us during MusicNet Login. There must also be a secure back office connection between MusicNet and your organization, over which we can return the token for validation in a server-to-server call. When authentication is successful, a userID (generated by you) is passed back to MusicNet.

This process assumes your organization manages secret information (e.g., symmetric keys) and is able to create encrypted tokens using standards algorithms (e.g., AES, 3DES, etc). The client receives this token during the normal login process to your service and then uses it again later to login to MusicNet. Upon receiving the token, we delegate the authentication back to your system over a secure connection. This can be a VPN connection established between MusicNet and your service, or it may be a stateless HTTPS requests-response originating from MusicNet. Since this method uses opaque tokens, it's unnecessary for us to know the structure or semantics of the tokens.

Your service then decrypts the token, validates the payload of the token, checks the authentication provider, and returns the userId if the authentication is successful.

This solution offers data integrity, confidentiality, non-repudiation, and is more secure against impersonation attacks. Usually tokens have an expiration time; beyond that time, the service discards the expired token and denies authentication. To impersonate valid users, malicious users would have to spoof these tokens in real time.

The obvious disadvantage of this solution is that the server-to-server call must take place over a far network link to delegate authentication. Potentially, this could reduce the scalability of the system. For this reason, we recommend implementing a process that enables us to validate tokens locally.

MusicNet Validates the Token Locally (Preferred)

Our preferred method of authentication requires the client to send a token that MusicNet can validate locally. If we know the secret key, and if we have the agreed upon structure and semantics for a token, we can decrypt the token and validate it locally. This solution is similar in concept to methods used by commercial distributed authentication services.

Local validation is superior to the delegated authentication method because server-to-server calls are unnecessary when tokens are validated locally.

However, this solution assumes that your organization can securely send a shared secret to us using an out-of-band process.

Eventually, we can design more complex key management processes using asymmetric cryptography or public key certificates.

Usually, the structure of the token follows this convention:

- ESK(salt || time stamp || userId)

- salt
  Random value
- time stamp
  Current UTC time
- userID
  Identity of user

The concatenation is encrypted with the shared secret key using a block cipher (e.g., AES, 3DES, or Blowfish).

Creating Daily Usage Reports

The music labels have reporting agreements that require MusicNet to gather the following information from all of our distribution partners:

- New Subscribers
- Renewed Subscribers
- Changes - Subscription Upgrades / Downgrades
- New Free Trials
- Canceled Subscribers
- Permanent Download Purchases (albums and tracks)

You'll need to generate a daily XML feed to report on these activities from the previous day's transactions.

This chapter describes how to construct XML feeds that satisfy music label requirements, how to create feeds that conform to our required format, and how to deliver those feeds to us.   It also provides an overview of the reconciliation process, which will be detailed more fully in an upcoming document.

Reporting Requirements

File Naming Convention

Please use this naming convention for your reports:

- RetailerID + DATE.xml
- RetailerID
  Unique retailer ID that MusicNet assigned to your organization.
- Date
  Reporting date (GMT): YYYYMMDD

Required Elements

The table below lists the elements that you must include in your XML feed for each reporting category.

- SUBSCRIBER_ID, ZIP_CODE, and TRANSACTION_DATE are required data elements for all reporting categories.
- AGE and GENDER are optional for all reporting categories.
- SUBSCRIBER_COUNT is required for all retailers offering subscriptions.
- STORE_COUNT is required for all retailers offering store accounts.

✓ **Note** All dates must be Greenwich Mean Time. Days of usage are GMT days from 12:00 A.M. through 11:59 P.M.

<Removed Table>

Delivery Methods

You can choose from two methods to deliver your daily usage reports:

You can create a web service on your network that allows us to download usage reports.

We recommend (and prefer) that you expose your daily usage reports via a Web service on your network; this enables us to automate the process of downloading and ingesting the report.

The Web service should expose an interface that enables us to request the data feed using a method call with the following signature:

string GetUserActivityData(string RetailerID, string Date).

- Required parameters:
- RetailerID
  string
- Date
  string (YYYYMMDD)

You can place your usage reports in a folder on the MusicNet SFTP server.

The other option for submitting daily usage reports is for you to copy your reports to a password protected folder that we'll set up for you on one of our SFTP servers. Your partner relationship representative will provide you with a user name and password for the directory. The SFTP server supports the SSH2 Protocol for secure file transfer.

You can download Gzip to compress the reports that you deliver to our SFTP server.

If you do compress your reports, please use the standard .gz file extension.

Note: To maintain security, and to ensure smooth delivery using this method, you must provide us with the range of IP addresses for systems that will access the SFTP site.

Data Feed Elements

ACCOUNT_TOTALS
Reports the total number of subscription accounts and the total number of store accounts.

STORE_COUNT
Total number of active accounts not associated with a subscription OFFER_SKU (referred to as "store accounts"). Users can purchase only permanent downloads through store accounts.

SUBSCRIBER_COUNT
Number of active accounts by OFFER_SKU – Total number of active accounts associated with each subscription OFFER_SKU offered by your service. Users may stream, subscription download, or permanent download MusicNet content.

<removed node desriptions>

Sample XML Data

```
<ACCOUNT_TOTALS>
<STORE_COUNT>
<COUNT>123</COUNT>
</STORE_COUNT>
<SUBSCRIBER_COUNT>
<COUNT>123</COUNT>
<OFFER_SKU>1111</OFFER_SKU>
</SUBSCRIBER_COUNT>
<SUBSCRIBER_COUNT>
<COUNT>123</COUNT>
<OFFER_SKU>2222</OFFER_SKU>
</SUBSCRIBER_COUNT>
</ACCOUNT_TOTALS>
```

<removed rest of data elements>

MusicNet XML Content Catalog

This guide describes the structure, elements, attributes and attribute types of the MusicNet Content Catalog Data Feed. The information here will help you better understand how you should implement the data feed in your particular environment. Because each platform and development environment is different, we’re unable to provide detailed steps for the actual implementation. Rather, this document gives you the background knowledge necessary for you to plan an implementation that's customized to your offerings.

Overview

The MusicNet Content Catalog Data Feed is an XML feed used to pass information to enable development of e-commerce applications and search tools hosted by MusicNet partners. This data describes tracks, albums, artists, metadata, rights, and wholesale prices for each item in the catalog. The structure of the feed illustrates the relationships between those elements.

Data Feed Concepts and Terminology

The following is a list of concepts and terminology, which are key to understanding how the MusicNet Content Catalog is designed:

Components
Logical object types of something that MusicNet offers, specifically albums, tracks, and videos. In other words, tracks, albums, and videos are all components. They are uniquely identified when associated with their corresponding data as defined in the Components element. There is a hierarchy between albums and tracks because several tracks are usually associated with one album. Videos are stand alone components.

Entity
We use the notion of an entity as a convenient way of naming components and artists.

In a larger context, an entity includes labels, labels owners, and retailers. These entities in a database will represent stand alone logical objects that have properties and relate one to another. Both hierarchies and dependencies exist between entities.

Price-Scopes
Defines price categories that distribution partners can use to help determine user retail prices.

SUBSCRIPTION_PRICE
The price distribution partners pay to MusicNet for components purchased by subscribers.

DMS_PRICE
The price distribution partners pay to MusicNet for components purchased by non-subscribers.

Artist-types
All Album components must include Artists, and all tracks must have acceptable Artist values, according to the list of Artist-types (e.g., Performer, Composer, Ensemble, etc.).

Primary and Foreign Keys
Most elements include primary key and foreign key attributes that make it possible to create relationships between elements.

Primary key
A unique identifier for each element.

Foreign key
A key in a table that refers to the primary key in another table.

Created-Date attribute
Applies to all elements in a feed except the Component-Parents element.

Last-Updated-Date attribute
The dynamic elements of data feeds include a Last-Updated-Date attribute.

The date format is CCYY-MM-DD
No left truncation is allowed.

Active-Status-Code attribute
Some entities include an Active-Status-Code attribute, which supports implementation of soft deletes.

Possible values are: A (active) and D (deactive).

ISRC
International Standard Recording Code. This is the international identification system for sound recordings and music video recordings. Each ISRC is a unique identifier for a specific recording, but the recording may appear on multiple albums. ISRCs are used to identify recordings for royalty payments and are often used by label partners as track-level identifiers.

ISRCs are used for reporting and track identification, and do not affect the process of playing, purchasing, or downloading songs.

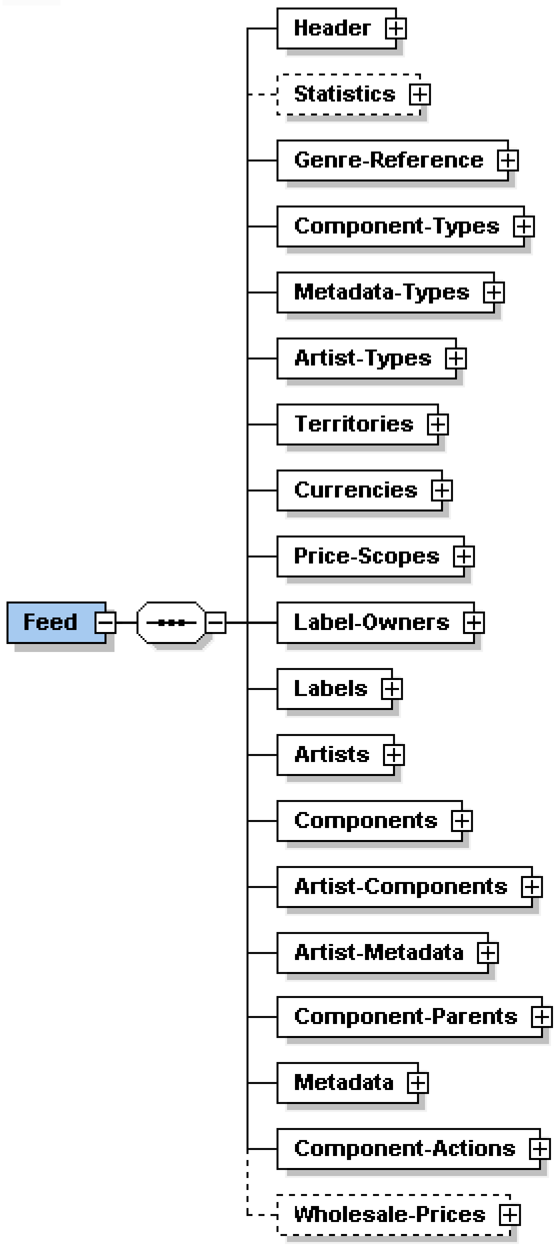
Digital ISRCs
Track-level digital identifiers (not actual ISRCs), generated by the Label partners and specific to their content.

Digital UPCs
Unique Product Codes used to identify parent components (i.e., albums or videos). Digital UPCs are generated by the Label partners to identify their digital products and to differentiate them from their physical products which use standard UPCs.

Structure of the Data Feed

Structure of the Data Feed

A root element ("Feed") and a sequence of children comprise the top level of the data feeds. All children elements (except the Header element) are organized to model SQL relations (tables). The following illustrates the relationships between these entities.



Relationships Between Artist, Album, and Track Components

There are numerous parent-child relationships between data feed components:

Artist_Components
Establishes a many-to-many relation between components and artists:

- Albums and tracks can have one or more artists;
- Artists can appear on one or more components.

Component_Parents
Establishes relationships between components:

- Albums have one or more tracks;
- Tracks belongs to one and only one album;
- The album of a track (in SQL lingo) is:

  select parent_comp_id from component_parents

  where child_comp_id = YourTrackId

Types of Information

Information in the data feed falls into two categories:

- **Static information (constant):** Information seldom updated or changed (e.g., metadata types). There is static information in all feeds, even if the information is unchanged.

  Note: Static information must be processed as an update.
- **Dynamic information**: Information routinely added and updated (e.g., tracks, albums, etc.).

Types of Distributable Content

There are two types of distributable content:

- **Retailer Exclusive Content**
  Content that is prepared exclusively for a MusicNet retail or distribution partner will be included in the respective partner's feed, and clearly identified with the exclusive indicator as defined in Components element.
- **Content Distributable to All Retailers**
  Retailers will receive data feeds with distributable content approved for all retailers, in addition to their retailer-exclusive content.

Types of Data Feeds

We publish both daily incremental data feeds and full data feeds (on request).

Full data feeds
Contain the entire active catalog of content that MusicNet offers.

Incremental data feeds
Contain updates that include new information and changes made since the last feed. We'll deliver these feeds to an agreed upon drop point according to your ingestion schedule, which should be at least every three days, if not daily.

Each feed will reside on your drop point for only 7 days.

Important When processing more than one incremental feed at a time, it's critical that you process the feeds in the order that they are created and only after you've established a baseline by processing a full feed.

If for some reason you're unable to process incremental feeds in the correct order, you can request a full feed from us, which you'll use to synchronize your catalog with ours. There are a variety of ways to ingest a full data feed without purging your entire content catalog each time. For example, you can process all components from the full feeds as update/inserts and mark the remaining components (if any) as deleted.

Due to the way the feeds are generated, updates may be hidden. Even though a feed may show that some components have been updated, it may look identical to the previous version. This happens when the updates were done for parts of the component not available in the feed. For example, when a new download file replaces an existing one, the feed snapshot image for the component will be the same (it had a download available before and after the update).

Data Feed Components

Incremental data feeds contain only distributable components unless previously active components must be deactivated. Full data feeds contain a snapshot of all distributable components.

Distributable components

Components that meet label and publisher approvals AND are approved for at least one of the following rights or actions:

• download in subscription;

• burn in subscription;

• stream in subscription;

• purchasable inside a subscription;

• or purchasable outside of subscription.

Albums

The distributable components rule also applies to albums, unless a track belonging to an album is exclusive to another partner. Albums appear in the feed with all their tracks if at least one track in the album is distributable.

Revoked components

A component Id is revoked from the data feed when after an update it has no available actions and thus is no longer distributable.

Components that are revoked appear in the next incremental feed with all actions listed as unavailable (set to 9999/12/31); a revoked component will not appear in subsequent incremental data feeds. You can choose to either delete a revoked component id completely or keep it in your database, so it's already in the system, should it become available again.

Soft deletes

A component Id is deleted from the data feed but that component still resides in the MusicNet content database. Due to reporting and platform constraints, the delete cannot be performed physically, but only as a logical (soft) delete.

Components that are soft deleted appear one last time in the next incremental feed with an indicator that shows you can remove it from your available catalog; the deleted component will not appear in subsequent incremental data feeds. It is up to you to decide if you're going to delete the component id completely or just perform your own soft delete.

Reactivated components

If a component's rights are restored, it will reappear in the next incremental feed.

<removed table>

Loading the Data Feed

While you're setting up a process to load your data feeds, we recommend that you consider the following suggestions:

Use a staging relational database

Because of their size, we recommend you load XML feeds into a staging relational database for rapid retrieval. You'll load lists of artists, lists of components with CD burn rights updated since a certain date, lists of albums into the corresponding database tables, etc.

Design your database schema to mirror the structure of the data feed

The data feed is indented to help convey the relationships between the elements in the feed. This makes it easier to design a relational database schema that matches the hierarchical structure of the feed.

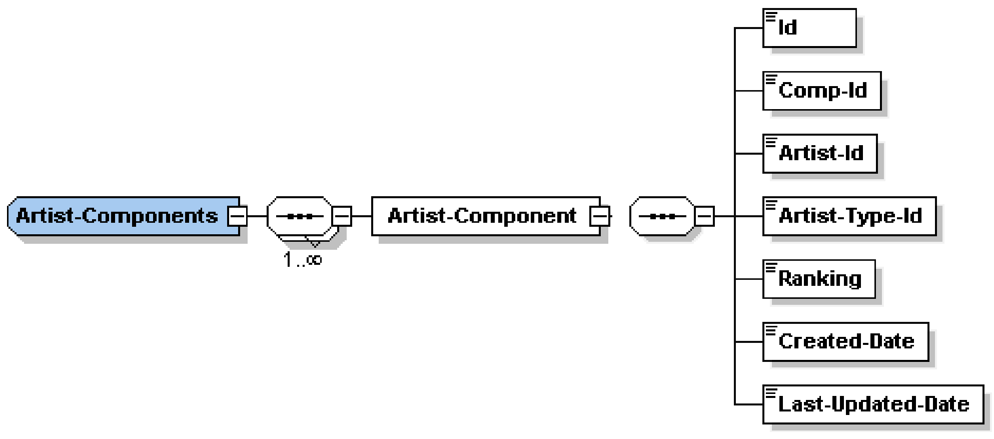
Make your implementation flexible

MusicNet occasionally adds elements or sub-elements to the data feed which changes the feed schema. Therefore, it's important that you carefully plan the implementation of your parsing code, so it can be easily adapted to support schema extensions.

Data Feed Elements and Attributes

Artist-Components

Defines the many-to-many relationship between artists and components. Defines the role a specific artist plays on that component.



<removed table>

Artists: The marketed artist(s) for an album is recorded in the MusicNet database at the album-level and is a required value.

Note: It's possible to have "Various Artists".

Performer: Lists the actual artists that perform on a track. The album-level Artist may not always appear as a Performer at the track level.

However, it's optional to have Performers at the track/child-level. Common acceptable Artist and Performer values:

- ARTIST (parent component value)
- DIRECTOR (parent component value)
- PERFORMER (child component/track Artist)
- COMPOSER
- CONDUCTOR
- SOLOIST
- ENSEMBLE
- ARRANGER

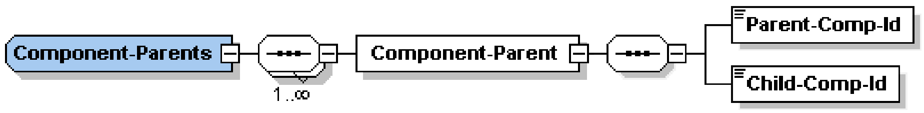
Note: It is possible to have "Various Artists" as an album-level artist value, but it should not appear as a track-level artist or performer value.

Sample XML Data

```
<Artist-Components>
    <Artist-Component>
    <Id>1046280</Id>
    <Comp-Id>1625113</Comp-Id>
    <Artist-Id>40681</Artist-Id>
    <Artist-Type-Id>1</Artist-Type-Id>
    <Ranking>1</Ranking>
    <Created-Date>2004-03-23</Created-Date>
    <Last-Updated-Date>2004-03-23</Last-Updated-Date>
    </Artist-Component>
</Artist-Components>
```

Component-Parents

Defines a hierarchy over components, which currently includes only albums and tracks.



<removed table>

Sample XML Data

```
<Component-Parents>
<Component-Parent>
<Parent-Comp-Id>1624571</Parent-Comp-Id>
<Child-Comp-Id>1624583</Child-Comp-Id>
</Component-Parent>
</Component-Parents>
```

<removed rest of data feed elements, additional data feed, availability scenarios, and relational schema>

Mass Import of Customer Valid Addresses

This chapter describes how to configure the Customer Valid Address Mass Import feature in IBS, which will enable you to upload valid addresses to the IBS database.

A valid address is an address record that you obtain from a reliable external source, such as a data file supplied by a postal service. The provider confirms that each address in the file is a valid delivery point. IBS stores these addresses in the VALIDADDRESS table of its database. You can use the mass import feature to import new valid addresses or to update the existing ones in this table.

✓ **Note** This feature supports only valid address data.

Requirements

The following list describes the requirements necessary to setup and use the IBS Valid Address Mass Import feature:

- The IBS Integration Windows Service must be installed and running.
- The Mass Import Windows service must be installed and running. The configuration settings are located inside of the configuration file, which is installed with IBS.
- Latest XML Template
- Latest XMP Map
- If your integration database has the same structure as what was installed with IBS, this process will work for you. The following tables must be present and configured with appropriate values based on your environment. These tables include:
  - DSN
  - ENDPOINT
  - ENDPOINT_TYPE
  - FILE_PATH
  - FILE_WATCHER
  - FTP_ENDPOINT
  - FTP_WATCHER
  - INTEGRATION_EVENT
  - LISTENER
  - WORKER_CONDITION
  - WORKER

Overview

In IBS, valid address import involves two processes. First, you will convert a flat file into an XML file, and then you will upload the XML file to the VALID_ADDRESS table in the IBS database.

Flat File to XML Conversion

You will begin the file transformation process by placing a flat file with valid addresses to be processed in a directory that you designate during the integration database configuration. IBS will convert the file to an XML format that can be uploaded directly to IBS.

1. The Integration Component uses an FTP_WATCHER to monitor a configured FTP directory for a flat file of a specified filetype (e.g., .txt, .csv). Filetype is stored in the FTP_WATCHER table.
2. Begin the file conversion by placing a flat file with customer address data in the FTP Directory that you specified in the integration database during configuration.
3. When a new flat file is detected by the FTP_WATCHER, it triggers the Integration Component to rename (adds a new prefix in a GUID like format) and remove the file from the FTP directory for processing.
4. The Integration Component maps the data from the flat file to the XML template using file structure rules that are defined by the XML map file.
5. The output of this process is an intermediary XML file that contains a collection of valid addresses that are ready to be uploaded to the IBS database.
6. The integration component stores the intermediary XML file containing the addresses in the holding directory that you configured in the integration database when you set up the process.

XML to Database Upload

The input file for this phase of the import process is the transformed XML output file that was generated by the integration component in previous steps.

7. Begin by moving the intermediary XML file (containing the valid addresses) from the holding directory (Step 6) to the upload directory. The input folder for this process is different from the output folder in the last step.

   You can automate this process, and eliminate the extra step of moving files, by making the output directory of the previous process the same as the input (file drop) directory for this process.

```
      <appSettings>
          <add key="ServiceLocatorUrl" value="http://eng-v6-03/asm/Development/
All/ServiceLocation.svc" />
          <add key="UserName" value="entriqeng" />
          <add key="Proof" value="entriqeng" />
          <add key="Dsn" value="unittest" />
          <add key="CheckForNewFilesIntervalSecs" value="5" />
          <add key="DirectoryToWatch" value="C:\IBS Interprit\Integration\MassIm-
port"/>
          <add key="CompletedFilesDirectory" value="C:\IBS Interprit\Integra-
tion\MassImport\Completed" />
          <add key="ConnectionString" value="user id=datarepository_dev;pass-
word=datarepository_dev;data source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTO-
COL=TCP)(HOST=ENG-ORADB-A)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=DEV)))"/>
      </appSettings>
```

8 A watcher that points to the upload directory detects the new file and triggers the upload process to begin.

9 The Mass Import Service removes the intermediary file from the upload directory and uploads the contents of the file to the IBS database. The processed file is renamed and moved to the CompletedFilesDirectory, the path to which is configured in the App.config file.

10 The Mass Import Service creates a new folder (named with the current date) within the upload directory and archives the uploaded file(s) in that folder.

11 The Mass Import Service logs a PASS/FAIL message in the SYSTEM_DATA table for each valid address record that is processed.

Updating Customer Data

The Customers Web Service, which uploads valid addresses to IBS, also performs several checks on the data prior to writing data to the database, including checks that verify:

- No duplicate Valid Addresses exists. If a duplicate address is found, the current record is skipped. If a duplicate valid addresses does not exist, the new valid address is written to the database.
- Changes to the ValidAddress table that affect the logic in a customer's record are propagated throughout the system. For example, IBS will find all customers with addresses linked to old valid address and update their fields. Figure 1 illustrates this process:

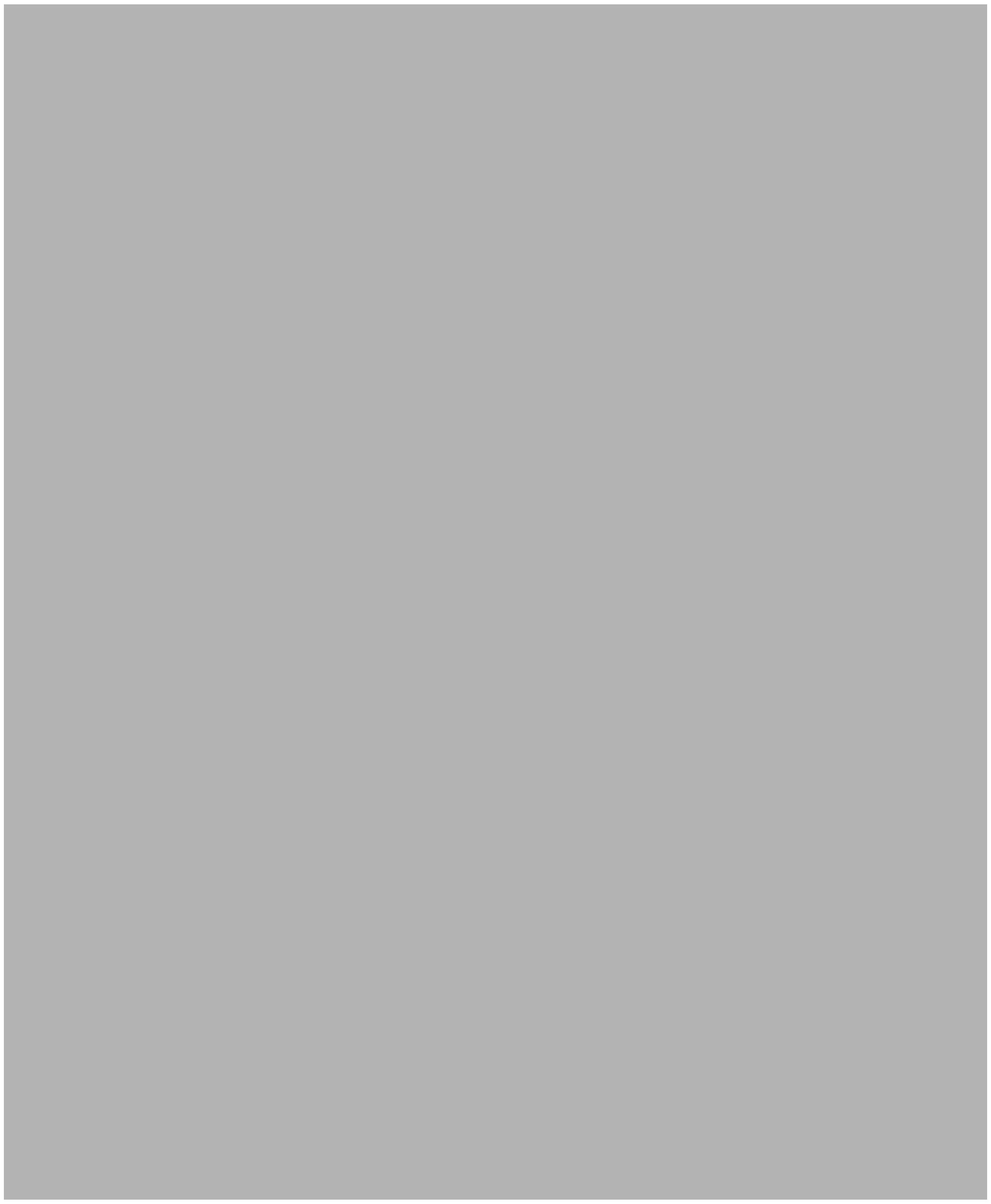


Figure 1: Import Overview

Configuration

IBS Integration Windows Service

The IBS Integration Windows Service controls the entire process of converting flat files to XML. It reads the FTP_WATCHER table to locate the watchers that must be started.

The IBS Integration Windows Service is built into the Integration Component, which is automatically installed with each IBS build. You must configure and then start this service prior to beginning the Valid Address Mass Import process.

To configure the IBS Integration Windows Service, you must change the connection string in its app.config file, so it points to the integration database that houses all of your configuration settings.

When you start the Windows Service, it will look for the FTP Directory that is stored in the FTP_ENDPOINT table of the database. If you require authenticated access to reach the FTP endpoint, you must change the username and password fields in the FTP_ENDPOINT table. The default values for these fields allows anonymous access to the directory.

Before running your Windows Service for the first time, you should manually access the FTP directory to verify that a connection exists at that location.

Windows Mass Import Service

The Windows Mass Import Service relies on a configuration file to retrieve the settings it uses to process and upload valid addresses.

For most installations, the file is located in the installation directory of the Mass Import Service. Depending on your installation, the configuration file will be named either app.cfg or WindowsService.exe.cfg.

✓ **Note** Future releases of the Integration Component will include a file configuration manager, which provides a GUI that simplifies the configuration of this file.

You can use any text editor to change settings in app.cfg or WindowsService.exe.cfg.

The following keys contain values that can vary according to your environment.

- ServiceLocatorUrl—Location of the Web service that updates and populates the IBS database.
- UserName—Username for directories that require authentication.
- Proof—Password for directories that require authentication.
- Dsn—Data Source Name required to specify the connection information for a database server.
- CheckForNewFilesIntervalSecs—Defines how often the Mass Import Service checks the input directory for new XML files.
- DirectoryToWatch—Path to the input directory.
- **CompletedFilesDirectory**—Location where the Mass Import Service is to store files that have been uploaded.

- **ConnectionString**—A string that specifies information about the data source and the means of connecting to it.

```
<appSettings>
  <add key="ServiceLocatorUrl" value="http://asm-srv/ServiceLocation.svc"/>
  <add key="UserName" value="entriqeng" />
  <add key="Proof"    value="entriqeng" />
  <add key="Dsn"      value="unittest" />
  <add key="CheckForNewFilesIntervalSecs" value="5" />
  <add key="DirectoryToWatch" value="C:\DirectoryPath" />
  <add key="CompletedFilesDirectory" value="C:\DirectoryPath" />
  <add key="ConnectionString" value="Oracle Connection String" />
</appSettings>
```

Integration Database Tables

You will store many of the configuration settings for the Windows and Mass Import services in the following integration database tables.

CONTROLLER

This table sets time intervals for checking FTP and file drop locations.

Table 2: CONTROLLER

Column Name	ID	Pk	Null?	Data Type	Default	Histogram
ID	1	1	N	NUMBER (9)		Yes
LISTENER_ID	2		N	NUMBER (9)		Yes
TYPENAME_ID	3		N	NUMBER (9)		Yes
POLLING_INTERVAL_MS	4		N	NUMBER (9)		Yes
MAX_THREAD_COUNT	5		Y	NUMBER (9)		Yes

If there is a WORKER_DATA_PROCESSOR view in database it is useful to view complete data for this process. If not, this script will show all information about Valid Address process:

```
select
  w.ID as "WORKER_ID", w.NAME as "WORKER_NAME",
  d.NAME as "DSN", ttn.TYPENAME as "TRANSFORMER_TYPENAME", fp.PATH as "TRANSFORMER_PATH",
fp.SPECIAL_FOLDER as "TF_SPECIAL_PATH",
  fe.*, pptn.TYPENAME as "POST_PROCESSOR_TYPENAME"
from WORKER w
join DATA_PROCESSOR dp on dp.WORKER_ID = w.ID
left join DSN d on d.ID = dp.DSN_ID
left join TYPENAME ttn on ttn.ID = dp.TRANSFORMER_TYPENAME_ID
left join FILE_PATH fp on fp.ID = dp.TRANSFORMER_MAP_PATH_ID
left join FIXED_ENDPOINT fe on fe.ENDPOINT_ID = dp.OUTBOUND_ENDPOINT_ID
left join TYPENAME pptn on pptn.ID = dp.POST_PROCESSOR_TYPENAME_ID
```

FIXED_ENDPOINT View is useful to check on endpoint configurations:

```
select
  -- Endpoint values.
```

```
   ep.ID as "ENDPOINT_ID", ep.NAME as "ENDPOINT_NAME", ep.ENDPOINT_TYPE_ID, eptn.TYPENAME as
"ENDPOINT_TYPENAME",
   wcf.ADDRESS as "WCF_ADDRESS", wcf.BINDING as "WCF_BINDING", wcf.CONTRACT as
"WCF_CONTRACT", wcf.BINDING_CONFIGURATION as "WCF_BINDING_CONFIGURATION",
   tcp.ADDRESS as "TCP_ADDRESS", tcp.PORT_NUMBER as "TCP_PORT",  tcp.DISCONNECT_AFTER_SEND as
"TCP_DISCONNECT_AFTER_SEND",  tcp.RELATED_MAILBOX_ID as "TCP_RELATED_MAILBOX_ID",
tcp.MAX_RETRY_ATTEMPTS as "TCP_MAX_RETRY_ATTEMPTS",  tcp.RECONNECT_DELAY_MS as
"TCP_RECONNECT_DELAY_MS",
   ftp.ADDRESS as "FTP_ADDRESS", ftp.USERNAME as "FTP_USERNAME", ftp.PASSWORD as
"FTP_PASSWORD", ftp.IS_SFTP as "FTP_IS_SFTP", ftp.TRANSFER_INTERVAL_MS as
"FTP_TRANSFER_INTERVAL_MS", ftp.TRANSIT_FILE_EXTENSION as "FTP_TRANSIT_FILE_EXTENSION",
   fp.FILE_TYPE as "PATH_FILE_TYPE", fp.PATH, fp.SPECIAL_FOLDER as "PATH_SPECIAL_FOLDER",

   -- Target worker values.
  wc.ID as "WORKER_CONDITION_ID", ct.NAME as "WORKER_CONDITION_NAME", wc.CONDITION_VALUE
as "WORKER_CONDITION_VALUE"
--   ,fp.TRANSFER_INTERVAL_MS AS "FILE_TRANSFER_INTERVAL_MS"

from ENDPOINT ep
left join TYPENAME eptn on eptn.ID = ep.TYPENAME_ID
left join WCF_ENDPOINT wcf on wcf.ID = ep.ENDPOINT_ENTITY_ID and ep.ENDPOINT_TYPE_ID = 1
left join TCP_ENDPOINT tcp on tcp.ID =  ep.ENDPOINT_ENTITY_ID and ep.ENDPOINT_TYPE_ID = 2
left join FTP_ENDPOINT ftp on ftp.ID = ep.ENDPOINT_ENTITY_ID and ep.ENDPOINT_TYPE_ID = 3
left join FILE_PATH fp on fp.ID = ep.ENDPOINT_ENTITY_ID and ep.ENDPOINT_TYPE_ID = 4
-- Target worker tables.
left join WORKER_CONDITION wc on wc.ID = ep.ENDPOINT_ENTITY_ID and ep.ENDPOINT_TYPE_ID = 5
left join CONDITION_TYPE ct on ct.ID = wc.CONDITION_TYPE_ID
```

DATA_PROCESSOR

This table hold references to processing worker and it's transformation and mapping path.

Table 3:

Column Name	ID	Pk	Null?	Data Type	Default	Histogram
ID	1	1	N	NUMBER (9)		Yes
WORKER_ID	2		N	NUMBER (9)		Yes
DSN_ID	3		Y	NUMBER (9)		Yes
TRANSFORMER_TYPENAME_ID	4		Y	NUMBER (9)		Yes
TRANSFORMER_MAP_PATH_ID	5		Y	NUMBER (9)		Yes
OUTBOUND_ENDPOINT_ID	6		N	NUMBER (9)		Yes
POST_PROCESSOR_TYPENAME_ID	7		Y	NUMBER (9)		Yes
TRANSFORMER_CODE_PAGE	8		Y	NUMBER (9)		Yes

DSN

The DSN table provides internal mapping for database connections used by numerous services.

Table 4: DSN

ID	NAME
2	DENMARK
5	FINLAND
4	NORWAY
3	SWEDEN

ENDPOINT

The ENDPOINT table represents the path to a directory with a specific endpoint type, which is defined in ENDPOINT_TYPE table (Table 6).

The TYPENAME_ID is a reference in the TYPENAME table that represents an assembly, which will be loaded by a process to execute desired functionality.

Table 5: ENDPOINT

ID	NAME	TYPENAME_ID	ENDPOINT_TYPE_ID	ENDPOINT_ENTITY_ID
10	AS_02 Output	24	3	15
11	AS_04 Output	22	4	18
1	EventListener	1	1	1
3	ProvisioningListener	1	1	2
6	FtpWatcher	8	6	
43	EventListeneR 1	79	1	1
45	ProvisioningListener	81	1	2

ENDPOINT_TYPE

The values in this table are default values and must *not* be changed.

Table 6: ENDPOINT_TYPE

ID	NAME
6	EmptyEndpoint
4	FilePathEndpoi nt
3	FtpEndpoint
5	ParentListener Endpoint
2	TcpEndpoint
1	WcfEndpoint

FILE_PATH

This table stores the filepath for the output of the intermediary XML file. In the filepath, you must add the path to the location where the file will be converted (e.g., IBS/

Downloads/AS04) each day. This is the output folder used by the file conversion proccess. By default, the converted file is placed in the same folder as the source file.

Table 7: FILE_PATH

ID	FILE_TYPE	PATH
24	output	C:\IBS Interprit\Integration\Output\3P30\MassExportPage.xml
16	map	Transformers\AddressSuppliers\AS_02_DenmarkMap.xml
1	xml	Transformers\AddressSuppliers\AS_04_Map.xml
18	output	\\integration\IBS\InBound\AddressSupplier\04\AS_04.xml
2		C:\IBS Interprit\Integration\Downloads\AS04

FILE_WATCHER

Table 8: FILE_WATCHER

ID	NAME	LISTENER_ID	FILE_PATH_ID	FILE_EXTENSION_LIST	STORAGE_FILE_PATH_ID	FORWARDING_ENDPOINT_ID	DSN_ID	WORKER_CONDITION_ID	IS_ENABLED
2	ImportEFakturaFileWatcher	169	7	*.txt	8	4	1	5	1
1	ImportAvtaleGiro-FileWatcher	169	18	*.txt	19	4	1	4	1

FTP_ENDPOINT

This table defines which port and ftp directory that the watcher monitors for the flat file with valid addresses. This is the pick-up location for the flat file. The FTP_ENDPOINT_ID for the FTP address listed here must be referenced in the FTP_WATCHER table.

Table 9: FTP_ENDPOINT

ID	ADDRESS	USERNAME	PASSWORD	IS_SFTP	TRANSFER_INTERVAL_MS	TRANSIT_FILE_EXTENSION
15	ftp://localhost/AS02Output/	anonymous	anonymous	0	300	
1	ftp://localhost/AS04Input/	anonymous	anonymous	0	1000	

FTP_WATCHER

You must create your FTP watcher last because it is the culmination of several other tables.

- Before installing the FTP_WATCHER, you must have a watcher installed. The location referenced by the listenerID must be a valid FTP site.
- You must also have an FTP endpoint before setting up the FTP watcher because the FTP Watcher will relate it to FTP endpoint. There must be a reference here.

Table 10: FTP_WATCHER

ID	NAME	LISTENER_ID	FTP_ENDPOINT_ID	FILE_EXTENSION_LIST	STORAGE_FILE_PATH_ID	FORWARDING_ENDPOINT_ID	DSN_ID	WORKER_CONDITION_ID	IS_ENABLED
1	AS_04AS_04	5	1	*.txt	2	2	6	1	1

INTEGRATION_EVENT

Table 11: INTEGRATION_EVENT

ID	NAME	DIRECTION	DESCRIPTION
105	RepeatEvent	out	This is a testing event which simply repeats its input as its output.
178	AS04FtpDownload	in	
150	PI01FtpDownload	in	

LISTENER

Table 12: LISTENER

ID	IC_ID	INBOUND_ENDPOINT_ID	SINGLETON_TYPENAME_ID	SETTINGS
16	1	47	91	
3	1	4	6	

WORKER

Specifies the nave of the executing worker process it's type and settings if any.

Table 13: WORKER

ID	NAME	TYPENAME_ID	SETTINGS	IC_ID
1	AS_04_UpdateCustomerData	19		

WORKER_CONDITION

Table 14: WORKER_CONDITION

ID	WORKER_ID	CONDITION_TYPE_ID	CONDITION_VALUE
86	58	3	44
88	45	1	455
84	56	9	Monthly export (print)

Mappings

The following table (Table 15) shows the mapping of the values from the XML input to a ValidAddress Object.

Here is a snippet from this file, which has position of 122 in a flat file and data length of 40 characters. This data should be placed into the template file with the following XPath:

p:Customer/p:DefaultAddress/p:Street

```
<Line>
      <MustEqual value="UA">
        <Substring startIndex="0"
                   length="2"/>
      </MustEqual>
      <Input>
        <Substring startIndex="122"
                   length="40"/>
      </Input>
      <Output path="p:Customer/p:DefaultAddress/p:Street"/>
</Line>
```

Table 15: XML Input Mapped to ValidAddress Object

ValidAddress Field	XML Source
ExternalReference	DefaultAddress/ExternalReference
CareOfName	DefaultAddress/CareOfName
Street	DefaultAddress/Street
HouseNumberAlpha	DefaultAddress/HouseNumberAlpha
HouseNumberNumeric	DefaultAddress/HouseNumberNumeric
Extra	DefaultAddress/Extra
SmallCity	DefaultAddress/SmallCity
BigCity	DefaultAddress/BigCity
PostalCode	DefaultAddress/PostalCode
Province	DefaultAddress/ProvinceKey
Country	DefaultAddress/CountryKey

The following table (Table 16) shows a mapping of values from the XML input to a ValidAddressCriteria object. Only apply values whose XML source is not nil.

✓ **Note** HouseNumberNumeric was updated to nil to avoid crashes if data is absent or not numeric.

Table 16: XML Input Mapped to ValidAddressCriteria Object

ValidAddressCriteria Field	XML Source
ExternalReference	DefaultAddress/ExternalReference
Street	DefaultAddress/Street
HouseNumberAlpha	DefaultAddress/HouseNumberAlpha
HouseNumberNumeric	DefaultAddress/HouseNumberNumeric
SmallCity	DefaultAddress/SmallCity
BigCity	DefaultAddress/BigCity
PostalCode	DefaultAddress/PostalCode
ProvinceKey	DefaultAddress/ProvinceKey
CountryKey	DefaultAddress/CountryKey

The following table (Table 17) shows a mapping of values to be updated on the Customer object. Only apply values whose XML source is *not* nil.

Table 17: Values to Be Updated on the Customer Object

Customer Field	Source
DefaultAddress.FirstName	DefaultAddress/FirstName
DefaultAddress.Surname	DefaultAddress/Surname
DefaultAddress.TitleKey	DefaultAddress/TitleKey
DefaultAddress.Email	DefaultAddress/EmailAddress
BirthDate	DateOfBirth
StatusKey	StatusKey
DefaultAddress.HomePhone	DefaultAddress/HomePhone
DefaultAddress.WorkPhone	DefaultAddress/WorkPhone
DefaultAddress.Fax1	DefaultAddress/Fax1
DefaultAddress.Fax2	DefaultAddress/Fax2
Directions	DefaultAddress/Directions
ReferenceNumber	ReferenceNumber
LanguageKey	LanguageKey
DefaultAddress.ValidAddressId	(From object) ValidAddress.Id

Address Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 (http://www.altova.com) by JL (ENTRIQ INC.) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qual-
ified" attributeFormDefault="unqualified">
  <xs:element name="AddressExport">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Customers">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Customer" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:choice>
                      <xs:element name="CustomerID" type="xs:positiveInteger"/>
                      <xs:element name="ReferenceNumber" type="xs:string"/>
                    </xs:choice>
                   <xs:element name="DateOfBirth" type="xs:date" nillable="true"/
>
                    <xs:element name="LanguageKey" type="xs:token" nill-
able="true"/>
                   <xs:element name="StatusKey" type="xs:token" nillable="true"/>
                    <xs:element name="DefaultAddress">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="ExternalReference" type="xs:string"/>
                          <xs:element name="FirstName" type="xs:string" nill-
able="true"/>
                          <xs:element name="Surname" type="xs:string" nill-
able="true"/>
                          <xs:element name="TitleKey" type="xs:string" nill-
able="true"/>
                          <xs:element name="EmailAddress" type="xs:string" nill-
able="true"/>
                          <xs:element name="HomePhone" type="xs:string" nill-
able="true"/>
                          <xs:element name="WorkPhone" type="xs:string" nill-
able="true"/>
                        <xs:element name="Fax1" type="xs:string" nillable="true"/
>
                        <xs:element name="Fax2" type="xs:string" nillable="true"/
>
                          <xs:element name="CareOfName" type="xs:string" nill-
able="true"/>
                          <xs:element name="Street" type="xs:string" nill-
able="true"/>
                          <xs:element name="HouseNumberNumeric" type="xs:string"
nillable="true"/>
                          <xs:element name="HouseNumberAlpha" type="xs:string"
nillable="true"/>
                          <xs:element name="Extra" type="xs:string"/>
                          <xs:element name="SmallCity" type="xs:string" nill-
able="true"/>
                          <xs:element name="BigCity" type="xs:string" nill-
able="true"/>
```

```
                                <xs:element name="PostalCode" type="xs:string" nill-
            able="true"/>
                                <xs:element name="ProvinceKey" type="xs:string" nill-
            able="true"/>
                                <xs:element name="CountryKey" type="xs:string" nill-
            able="true"/>
                                <xs:element name="Directions" type="xs:string" nill-
            able="true"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="createDate" type="xs:date" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:schema>
```

AS04 Template

```
            -
            <CustomerContainerCollection xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
            xmlns="MassImport">
              - <Items>
                - <CustomerContainer>
                  - <Customer>
                      <BirthDate>1979-12-04T00:00:00</BirthDate>
                      <BusinessUnitId i:nil="true" />
                      <ClassId i:nil="true" />
                      <CustomFields i:nil="true" />
                    - <DefaultAddress>
                        <BigCity />
                        <CareOfName />
                        <CountryKey i:nil="true" />
                        <Directions i:nil="true" />
                        <Email i:nil="true" />
                        <Extended i:nil="true" />
                        <Extra i:nil="true" />
                        <Extra1 i:nil="true" />
                        <Extra2 i:nil="true" />
                        <Extra3 i:nil="true" />
                        <Extra4 i:nil="true" />
                        <Extra5 i:nil="true" />
                        <Fax1 i:nil="true" />
                        <Fax2 i:nil="true" />
                        <FirstName />
                        <GeoCodeId i:nil="true" />
                        <HomePhone i:nil="true" />
                        <HouseNumberAlpha i:nil="true" />
                        <HouseNumberNumeric i:nil="true" />
                        <Id i:nil="true" />
                        <MarketSegmentId i:nil="true" />
```

```
        <PostalCode />
        <ProvinceKey i:nil="true" />
        <SmallCity i:nil="true" />
        <Street />
        <Surname />
        <TitleKey i:nil="true" />
        <ValidAddressId i:nil="true" />
        <WorkPhone i:nil="true" />
      </DefaultAddress>
      <EmailNotifyOptionKey i:nil="true" />
      <ExemptionCodeKey i:nil="true" />
      <ExemptionFrom i:nil="true" />
      <ExemptionSerialNumber i:nil="true" />
      <Extended i:nil="true" />
    - <FinancialAccounts>
        <Items />
        <More>false</More>
        <Page>0</Page>
      </FinancialAccounts>
      <FiscalCode i:nil="true" />
      <FiscalNumber i:nil="true" />
      <Id />
      <InternetPassword i:nil="true" />
      <InternetUserId i:nil="true" />
      <IsDistributor i:nil="true" />
      <IsHeadend i:nil="true" />
      <IsProductProvider i:nil="true" />
      <IsServiceProvider i:nil="true" />
      <IsStockHandler i:nil="true" />
    - <Keywords>
        <Items />
        <More>false</More>
        <Page>0</Page>
      </Keywords>
      <LanguageKey i:nil="true" />
      <Magazines i:nil="true" />
      <PreferredContactMethodId i:nil="true" />
      <ReasonAddressId>0</ReasonAddressId>
      <ReasonId>0</ReasonId>
      <ReasonStatusId>0</ReasonStatusId>
      <ReasonTypeId>0</ReasonTypeId>
      <ReferenceNumber i:nil="true" />
      <ReferenceTypeKey i:nil="true" />
      <SegmentationKey i:nil="true" />
      <StatusKey i:nil="true" />
      <TypeKey i:nil="true" />
      <XmlDataDate />
    </Customer>
  - <ValidAddress>
      <Active>true</Active>
      <BigCity />
      <CareOfName>Ryan Flohr</CareOfName>
      <Country i:nil="true" />
      <CreateDateTime i:nil="true" />
      <Extended i:nil="true" />
      <ExternalAddressId i:nil="true" />
      <ExternalReference i:nil="true" />
```

```
        <Extra />
        <FromNumber i:nil="true" />
        <Gps />
        <HouseNumberAlpha i:nil="true" />
        <HouseNumberNumeric i:nil="true" />
        <Id i:nil="true" />
        <LongString />
        <ManuallyAdded i:nil="true" />
        <OddEvenBoth i:nil="true" />
        <PostalCode />
        <Province i:nil="true" />
        <SmallCity />
        <Street />
        <ToNumber i:nil="true" />
        <UniqueAddress i:nil="true" />
      </ValidAddress>
    </CustomerContainer>
  </Items>
  <More>false</More>
  <Page>0</Page>
</CustomerContainerCollection>
```

AS04 Map

Do not use Default Address for Valid Address import.

```
<?xml version="1.0" encoding="utf-8" ?>
- <!--
-->
- <FileMap>
  <TemplatePath>Transformers\AddressSuppliers\AS_04_Template.xml</TemplatePath>
  <NamespacePrefix>p</NamespacePrefix>
  - <DetailLines path="p:CustomerContainerCollection/p:Items/p:CustomerCon-
tainer">
    - <Line>
      - <MustEqual value="UA">
        <Substring startIndex="0" length="2" />
      </MustEqual>
      - <Input>
        <Substring startIndex="2" length="20" />
      </Input>
      <Output path="p:Customer/p:Id" />
    </Line>
    - <Line>
      - <MustEqual value="UA">
        <Substring startIndex="0" length="2" />
      </MustEqual>
      - <Input>
        <Substring startIndex="122" length="40" />
      </Input>
      <Output path="p:ValidAddress/p:Street" />
    </Line>
    - <Line>
      - <MustEqual value="UA">
        <Substring startIndex="0" length="2" />
      </MustEqual>
      - <Input>
      <Substring startIndex="162" length="5" />
    </Input>
    <Output path="p:ValidAddress/p:HouseNumberNumeric" />
  </Line>
  - <Line>
    - <MustEqual value="UA">
      <Substring startIndex="0" length="2" />
    </MustEqual>
    - <Input>
      <Substring startIndex="167" length="1" />
    </Input>
    <Output path="p:ValidAddress/p:HouseNumberAlpha" />
  </Line>
  - <Line>
    - <MustEqual value="UA">
      <Substring startIndex="0" length="2" />
    </MustEqual>
    - <Input>
      <Substring startIndex="174" length="34" />
    </Input>
    <Output path="p:ValidAddress/p:BigCity" />
  </Line>
  - <Line>
    - <MustEqual value="UA">
      <Substring startIndex="0" length="2" />
    </MustEqual>
    - <Input>
      <Substring startIndex="208" length="4" />
    </Input>
    <Output path="p:ValidAddress/p:PostalCode" />
  </Line>
  - <Line>
    - <MustEqual value="UA">
      <Substring startIndex="0" length="2" />
    </MustEqual>
    - <Input>
      <Substring startIndex="232" length="34" />
    </Input>
    <Output path="p:ValidAddress/p:CareOfName" />
  </Line>
  - <Line>
    - <MustEqual value="UA">
      <Substring startIndex="0" length="2" />
    </MustEqual>
    - <Input>
      <Substring startIndex="401" length="4" />
    </Input>
    <Output path="p:Customer/p:XmlDataDate" />
  </Line>
</DetailLines>
</FileMap>
```

Administrator Guides

Some of the writing samples in this section are rough approximations of the actual documentation that I published. Several of my samples are publicly available, but in some cases, I had to remove illustrations, product names, company names, tables, and entire sections, so the copyright owner would grant me permission to use excerpts in my portfolio.

- <REMOVED> Administrator's Guide
- Dataflow Configuration Workstation

<REMOVED> Administrator’s Guide

The following topics are included in this chapter:

- <REMOVED>
- Intelligent Data Manager Components

<REMOVED>

The IBS is a multi-product <REMOVED> system designed to meet your credit application decisioning needs.

At this time, IBS offers two products:

- Capstone Decision Accelerator—A comprehensive rules and decision management system designed specifically for new account <REMOVED>s.
- Capstone Intelligent Data Manager—An advanced connectivity and data management system that provides pre-developed interfaces to consumer credit reporting agencies, business reporting agencies, and to the <REMOVED>® Expansion score service.

Each component functions as a stand-alone product; however, when used together, Decision Accelerator and Intelligent Data Manager provide a unified solution to support your business needs.

Intelligent Data Manager

Intelligent Data Manager provides connectivity and data management capabilities that can be used in conjunction with strategy and decision management, analytics, and application processing systems in order to provide a complete application processing solution.

Intelligent Data Manager provides the following functions:

- Manages connectivity for physical and network interfaces to external data sources including the three major US consumer credit reporting agencies (Equifax, Experian, and TransUnion), the two major Canadian consumer credit reporting agencies (Equifax Canada and TransUnion Canada), the major US business bureaus (D&B and Experian Business), Equifax SBFE, and the <REMOVED>® Expansion score service. Data source availability is determined by your Intelligent Data Manager contract.
- Accepts data through on-demand inquiries from your upstream <REMOVED> application(s) and constructs and submits inquiries to requested data sources.
- Collects retrieved data from data sources, manages data manipulation required, and prepares data for subsequent use by downstream applications.
- Provides product administration enabling technical users to configure subscriber numbers and view data source connectivity status for the product.
- Accepts requests directly from Decision Accelerator (if installed) and passes data back to Decision Accelerator for use in application decisioning.
- Allows you to make joint consumer credit report inquiries for an applicant and a co-applicant.

Using Intelligent Data Manager with Decision Accelerator

When Intelligent Data Manager and Decision Accelerator are installed together, Decision Accelerator becomes the calling system for Intelligent Data Manager. Using Decision Accelerator, you can define a decision flow that includes one or more Web service requests to Intelligent Data Manager for data from a US or Canadian credit reporting agency (CRA), business bureau, Equifax SBFE, or the <REMOVED>® Expansion score service, as well as ARF input. You can also define decision flows to implement business policies from externally obtained data, using data methods, rulesets, decision trees, decision tables and external models that use US and Canadian consumer credit report data and associated calculations.

Process Server

The <REMOVED>® Process Server is the main entry point for the <REMOVED> and is used to control processing for Intelligent Data Manager or both Intelligent Data Manager and Decision Accelerator, if you are using both products. As a runtime component, the Process Server accommodates a 24x7 availability window, except for scheduled downtime.

The main purpose of the Process Server is to route a call seamlessly to either Decision Accelerator or Intelligent Data Manager and to return a response to the calling application in XML format, including a summarized list of warnings and errors encountered during processing. The XML interface of the Process Server allows for easy integration with your calling system.

✓ **Note** If Decision Accelerator is also implemented, the Process Server that is installed during the Decision Accelerator installation process must be used.

Intelligent Data Manager Components

Intelligent Data Manager includes the following components:

- Dataflow Manager
- Connection Manager
- Dataflow Configuration Workstation
- Connection Configuration Workstation
- Data Storage Service
- Repositories

Dataflow Manager

Dataflow Manager is a *Web service* that manages the process of acquiring, parsing, normalizing, and analyzing data from a licensed third-party data source. It accepts data from your upstream <REMOVED> application, determines which data source should be accessed for each inquiry, dynamically builds the appropriate inquiry for each data source, and then submits the inquiries. It collects data retrieved from data sources, manages the required data manipulation and analysis, and prepares it for subsequent use by your calling application.

When using Intelligent Data Manager alone, your calling system submits requests to Intelligent Data Manager through the Process Server.When using Intelligent Data Manager with Decision Accelerator, Decision Accelerator can submit requests to Intelligent Data Manager directly

In each request submitted to Intelligent Data Manager, the calling system specifies which dataflow to execute from the following list of available dataflow options:

- Credit Reporting Agency Serial Dataflow
- Credit Reporting Agency Parallel Dataflow
- Credit Reporting Agency ARF Input Dataflow
- <REMOVED>® Expansion Score Dataflow
- Business Bureau Dataflow
- Business Bureau ARF Input Dataflow
- Equifax SBFE Dataflow

Credit Reporting Agency Serial Dataflow

The *CRA* Serial dataflow can acquire, analyze, and parse consumer credit reports from one or more of the CRAs for a given request. User–configurable rules determine whether more than one CRA is accessed. The order in which the CRAs are accessed for a given request is specified by the order in which the request nodes are supplied in the input message. For each data source of each applicant, the Intelligent Data Manager response can optionally contain the *ARF* data, the *parsed ARF*, the normalized ARF (*NARF*) data, consumer credit report analysis data, consumer credit bureau scores, and common credit bureau report images. Whether these items are returned in the Intelligent Data Manager response depends upon whether or not the calling system requested them.

Credit Reporting Agency Parallel Dataflow

The CRA parallel dataflow obtains credit reports from all data sources specified for a given request. This process can include acquiring, analyzing, and/or parsing consumer credit reports for each request. In other words, Intelligent Data Manager can process a single request to multiple CRAs simultaneously. For each data source of each applicant, the Intelligent Data Manager response can optionally contain the ARF data, the parsed ARF, the normalized ARF (NARF) data, consumer credit report analysis data, consumer credit bureau scores, and common credit bureau report images. Whether these items are returned in the Intelligent Data Manager response depends upon whether or not the calling system requested them.

Credit Reporting Agency ARF Input Dataflow

The ARF information is passed in as part of the XML message from a CRA request made by the client and can be sent to Intelligent Data Manager for additional processing. For each data source of each applicant, the Intelligent Data Manager response can optionally contain the raw ARF, the parsed ARF, the NARF, consumer credit report analysis data, consumer credit bureau scores, and a report image. Whether these items are returned in the Intelligent Data Manager response depends upon whether or not the calling system requested them.

<REMOVED>® Expansion Score Dataflow

The *FICO® Expansion score* (FES) dataflow acquires and analyzes FES data for a given request. Unlike the CRA dataflows that only support individual reports, all applicant data (up to two applicants) per request can be supplied in a single call to the <REMOVED>® Expansion score service. If the calling system wants to process two applicants jointly, it can call Intelligent Data Manager with a single request containing two applicants. If the calling system wants to process them separately, it can call Intelligent Data Manager with two requests, each containing a single applicant.

The Intelligent Data Manager response can optionally contain the entire response received from the <REMOVED>® Expansion score service, analysis data (score, consumer statements and reason codes), and a report image. Whether these items are returned in the Intelligent Data Manager response depends upon whether or not the calling system requested them.

Business Bureau Dataflow

For each request, the business bureau dataflow acquires and analyzes business data or obtains a list of similars (LOS) from either Experian Business or D&B. The dataflow will return a business data response and a credit risk score or a list of similars (LOS) for each request. Optionally, D&B Small Business Risk Insight (SBRI) data or Experian Business Small Business Intelliscore (SBI) data may be requested on input.

An LOS response is provided if a business ID is not known for a business and the information provided for that business matches several businesses at the bureau. The bureau returns up to 25 relevant businesses in a LOS indicating how well each business in the list matched the provided information by assigning each business in the list a score. The desired business can then be selected from the list of similars, and using the provided business ID, a request for a business data report can be submitted.

The Intelligent Data Manager response can optionally contain the raw ARF, business credit report, analysis data score, and a report image. Whether these items are returned in the Intelligent Data Manager response depends upon whether or not the calling system requested them.

For each request, the response can optionally contain the ARF, the analysis data, the scoring data, and the report image. Each of these is only returned when requested by the calling system.

Business Bureau ARF Input Dataflow

The ARF information is passed in as part of the XML message from a request made by the client. For each data source of each applicant, the Intelligent Data Manager response can optionally contain the raw ARF, business credit report analysis data, scores, and a report image. Optionally, D&B Small Business Risk Insight (SBRI) ARF data or Experian Business Small Business Intelliscore (SBI) ARF data may be requested on input.

Whether these items are returned in the Intelligent Data Manager response depends upon whether or not the calling system requested them.

Equifax SBFE Dataflow

The Equifax Small Business Financial Exchange (SBFE) Data dataflow acquires Equifax SBFE data for each request made by Intelligent Data Manager.

No analysis or parsing is performed on the response from Equifax; the response from Equifax is returned as XML.

Connection Manager

The Connection Manager Web service handles all connectivity functions for the Intelligent Data Manager product. It monitors physical and network interfaces to third-party data sources, including the CRAs and business bureaus. Your system application administrator configures the physical and logical links for the bureaus. Physical links are the actual physical lines between the client and the data source, such as VPN or a dedicated line. Logical links are the data source links in which requests are queued for processing from the CRAs and business bureaus, and other external data sources.

Using Intelligent Data Manager, you can request and receive data from any or all of the external data sources. When requesting data, you may also indicate which type of output is returned:

- **Scoring Data**—consumer credit bureau scores from CRAs and business credit risk scores from business bureaus.
- **Analysis Data**—credit analysis characteristics to be generated from consumer and business credit report data; for the FICO® Expansion score service, it includes FICO® Expansion scores, consumer statements, and exclusion reasons.
- **Report Images**—consumer credit reports, business credit reports, list of similars reports, and FICO® Expansion score reports. FICO® Expansion score report images are in PDF format and require extraction from the XML payload; all other reports are in text format and require reformatting for optimal viewing.
- **Common Report Images**—consumer credit report data in Fair Isaac's common format for all CRAs. The common report image is based on the normalized data used to generate analysis characteristics; it is in text format and requires reformatting for optimal viewing.
- **Raw ARF Data**—raw ARF (Automated Response Format) consumer credit report data from CRAs and list of similars or business credit report data from business bureaus.
- **Other Raw Data**—raw FICO® Expansion score transaction data and raw SBFE data.
- **Parsed ARF**—consumer credit report data from the CRAs, parsed by Intelligent Data Manager into XML elements and attributes.
- **Normalized ARF**—consumer credit report data that has been normalized across CRAs by Intelligent Data Manager.

If you have acquired CRA or business data using another application, Intelligent Data Manager can be used to process the ARF

Dataflow Configuration Workstation

The Dataflow Configuration Workstation (DCW) is a graphical interface that is used to configure subscriber numbers and other data source parameters for the data sources used. For more information on the DCW, see Chapter 2, "Dataflow Management."

Connection Configuration Workstation

The Connection Configuration Workstation (CCW) is a Microsoft Management Console (MMC) snap-in that runs on the Connection Manager host. The CCW is the graphical user interface used by the application administrator to set up the logical and physical links to external data sources. Physical links are the actual physical lines between the client and the data source, such as VPN or a dedicated line. The logical links determine how requests are queued for processing. The CCW resides on the same machine as the Connection Manager. If ARF data is submitted for *all* data sources, this component is not used. For more information on the CCW, see Chapter 3, "Connection Management."

Data Storage Service

The Data Storage Service (*DSS*) is an optional Web service that provides persistence of data to the *data repository*. The DSS stores processing history information received from Intelligent Data Manager to feed database tables used in administrative reports.

The DSS is only installed if administrative reports will be needed. For more information on DSS and administrative reports, see Chapter 4, "Data Storage Service and Reports."

Repositories

There are two databases that store information for Intelligent Data Manager:

- The Intelligent Data Manager repository for persisting the data source communication parameters
- The Message database for logging warnings and error messages

Intelligent Data Manager Workflow

The following is an overview of the Intelligent Data Manager workflow.

1. The application administrator uses the CCW to configure the logical and physical links for external data sources.
2. The application administrator uses the DCW to configure the data source subscriber numbers and data source parameter levels.
3. The calling system submits a transaction to the Process Server to process a data request in XML format.
4. The Process Server submits the transaction to the Dataflow Manager to execute the dataflow specified in the transaction.
5. The Dataflow Manager directs the Connection Manager to acquire the data from the data source(s).

6 The Connection Manager retrieves configuration settings for the bureau connections from the Intelligent Data Manager repository.

7 The Connection Manager sends a request to the specified data source and receives results.

8 The Connection Manager returns results to the Dataflow Manager.

Note Step 5 - step 8 are bypassed if the request is for ARF input.

9 The Dataflow Manager logs warning and error messages to the Intelligent Data Manager message database.

10 The Dataflow Manager returns data to the DSS for persistence to the data repository for use in reporting.

11 The DSS provides data used for viewing administrative reports with the Report Viewer.

12 The Dataflow Manager returns request results to the Process Server.

13 The Process Server returns request results to the calling system.

Deployment Options

Intelligent Data Manager can be used with or without other IBS products, such as Decision Accelerator. The sections that follow provide information on using Intelligent Data Manager by itself or with Decision Accelerator.

Using Intelligent Data Manager Only

<REMOVED> may only be deployed as a Web service.

The Process Server can be deployed as a .NET service on an IIS Server using Windows 2003.

Figure 2 depicts Intelligent Data Manager workflow in a .NET deployment.

Figure 2: Components of an Intelligent Data Manager in a .NET deployment

Using Intelligent Data Manager with other IBS Products

If you use Intelligent Data Manager with Decision Accelerator, your setup for Intelligent Data Manager will vary slightly from what is described in the previous section. The Process Server will be installed during the Decision Accelerator installation process and can hosted on the same server as the Strategy Management Workstation (SMW) or a separate host. Transactions between a calling application and the Capstone Origination Suite product always go through the Process Server.

Web Service Environment Setup

When using Intelligent Data Manager and Decision Accelerator in a Web service environment, the physical components consist of the following:

- **Client Host**—hosts a Web browser through which the risk management analyst accesses the SMW to author and update the rules used for decisioning. Also hosts the STF, a utility for testing newly-developed strategies created on the SMW, and an associated test data repository.
- **Process Server/SMW Host**—hosts the following components:
  - SMW Web application and the configuration repository, which use the host's native file system. The SMW Web application allows the editing of strategy configuration items in a Web browser interface. The configuration stores the rulesets, tables, trees, data methods, score models, scenarios, and decision flows authored in the SMW.
  - The Decision Accelerator runtime Web server, which provides the operating environment for the Process Server, Data Storage Service, and Report Viewer
  - The Process Server, which provides a Web service interface to the calling application (Web service client).
  - The Data Storage Service (DSS), which is deployed as a Web service and provides persistence of transaction and processing history XML data to the data repository for use in standard reports. It also stores a percentage of requests received at runtime by Decision Accelerator, in order to provide data to the STF.
  - The data repository, which resides within a relational database.
  - The *Report Viewer* Web application, which can access data collected by the DSS.

  Supported operating environments for the Process Server/SMW host include:
  - Windows or UNIX running Apache Tomcat
  - Windows or UNIX running an enterprise application server such as IBM *WebSphere*

✓ **Note** The SMW, Process Server, and Report Viewer can be hosted on separate application servers if desired.

- **Calling Application Host**—hosts the calling application (Web service client). The calling application is responsible for interacting with the external world, formulating Web service requests, sending them to the Process Server Web service, receiving responses, and processing them. The host machine can run any operating system that will support creation of Web service requests.
- **Intelligent Data Manager Server Host**—hosts the following components:
  - The Intelligent Data Manager Dataflow Manager, Dataflow Configuration Workstation, Connection Manager, Connection Configuration Workstation, and Intelligent Data Manager configuration repository.

  Supported operating environment for the Intelligent Data Manager server host: Windows running IIS.

✓ **Note** The Process Server and Apache Tomcat Web server can be hosted on separate application servers if desired.

Figure 3 shows the relationships between rule authors, system administrators, the SMW, Process Server, DSS, Report Viewer, Strategy Testing Facility, repositories, and calling application for the Web service deployment option for Decision Accelerator with Intelligent Data Manager.

Figure 3: Components of a Web service deployment with a combined Process Server/SMW host when using both Decision Accelerator and Intelligent Data Manager

EJB Environment Setup

When using Intelligent Data Manager and Decision Accelerator in an EJB environment, the physical components consist of the following:

- **Client Host**—hosts a Web browser through which the risk management analyst accesses the SMW to author and update the rules used for decisioning. Also hosts the STF, a utility for testing newly-developed strategies created on the SMW, and an associated test data repository.
- **Process Server/SMW Host**—hosts the following components:
  - The SMW Web application and the configuration repository, which use the host's native file system. The SMW Web application allows the editing of strategy configuration items in a Web browser interface. The configuration stores the rulesets, tables, trees, data methods, score models, scenarios, and decision flows authored in the SMW.
  - The Decision Accelerator runtime J2EE app server, which provides the operating environment for the Process Server, Data Storage Service, and Report Viewer.
  - The Process Server *stateless session bean*, which provides the Process Server EJB interface to the calling application (EJB client).
  - The Data Storage Service (DSS), which is deployed as a Web service and provides persistence of transaction and processing history XML data to the data repository for use in standard reports. It also stores a percentage of requests received at runtime by Decision Accelerator, in order to provide data to the STF.
  - The data repository, which resides within a relational database.
  - The Report Viewer Web application, which can access data collected by the DSS.

  Supported operating environments for the Process Server/SMW host include: Windows or UNIX running an enterprise application server such as IBM WebSphere.

  Note The SMW, Process Server, and Report Viewer can be hosted on separate application servers if desired.

- **Calling Application Host**—hosts the calling application (EJB client). The calling application is responsible for interacting with the external world, formulating stateless session bean client requests, sending them to the Process Server bean, receiving responses, and processing them. The calling application makes calls to the EJB through RMI. The host machine can run any operating system that will support creation of EJB requests.

- **Intelligent Data Manager Server Host**—hosts the following components:
  - The Intelligent Data Manager Dataflow Manager, Dataflow Configuration Workstation, Connection Manager, Connection Configuration Workstation, and Intelligent Data Manager configuration repository.

  Supported operating environment for the Intelligent Data Manager server host: Windows running IIS.

Figure 4 shows the relationships between rule authors, system administrators, the SMW, Process Server, DSS, Report Viewer, Strategy Testing Facility, repositories, and calling application for the EJB deployment option for Decision Accelerator with Intelligent Data Manager.

Figure 4: Components of an EJB deployment with a combined Process Server/SMW host when using both Decision Accelerator and Intelligent Data Manager

See Also For more information on implementation, see the *Capstone Intelligent Data Manager Getting Started Guide* and the *Capstone Intelligent Data Manager Developer's Guide*.

Dataflow Configuration Workstation

Dataflow within the Intelligent Data Manager is controlled by the Dataflow Manager. The Dataflow Configuration Workstation (DCW) is used to maintain data source parameters. The parameters are used to access and process data for data sources which currently include the US CRAs (Experian, TransUnion, and Equifax), Canadian CRAs (Equifax Canada and TransUnion Canada), the business bureaus (Experian Business and D&B), Equifax SBFE, and the FICO® Expansion score service.

See Also For more information on the Dataflow Manager and the available dataflow options, see "Dataflow Manager" on page 15.

This chapter covers the following topics:

Inquiry and Analysis Parameters

Inquiry and analysis parameters are contained in Parameters.xml files that are placed on your system during the installation process. Inquiry parameters (including subscriber numbers) control how Intelligent Data Manager constructs the inquiry to the external data source. Analysis parameters control how Intelligent Data Manager analyzes the data.

✓ **Note** Analysis parameters are not available from all data sources.

The Parameters.xml files contain all the default values for these parameters. During a transaction, if a parameter is not passed in, Intelligent Data Manager uses the default value from the file. Default parameter values are listed in the IBS data model spreadsheet. For more information on this spreadsheet, see the *Capstone Intelligent Data Manager Developer's Guide*.

For more information on the values contained in these files, or to modify the default values, contact your Fair Isaac account representative.

✓ **Note** Parameters that are passed in from your calling system will supersede the parameters that are entered in the DCW.

➡ **See Also** For information about acquiring subscriber numbers, see the *Capstone Intelligent Data Manager Getting Started Guide*.

Data Source Parameters

Data sources are the information sources from which you want to request data. Each data source has its own set of modifiable default parameters. These parameters can be organized to be applied in a hierarchical fashion during processing by assigning different values to selected parameters at different levels. A levels hierarchy can be four-deep, where the number of levels at each depth is unlimited. During processing, the precedence for applying parameters is listed as follows, starting with the highest priority:

1. Transaction input.
2. Level 4 parameters.
3. Level 3 parameters.
4. Level 2 parameters.
5. Level 1 parameters.
6. Data source defaults.

As part of a your organization, some business groups or product lines may have a need to regularly deviate from the parameter defaults established for the larger organization. You can use the DCW to create the needed variations through the administration of parameter levels. For example, some CRAs use different subscriber numbers to authorize users to retrieve elective data (such as specialized risk scores). You could use the DCW to set up different subscriber numbers for each product line or region, but still take advantage of other established parameter defaults. While this is an example of a two level hierarchy, the system can support up to four levels.

Launching the DCW

Use the instructions that follow to launch the DCW.

To launch the DCW

▶ From the **Start** menu select **Programs > <REMOVED> <REMOVED> <REMOVED> > Dataflow Configuration Workstation**.

Understanding the DCW

The sections that follow describe the components found in the DCW interface including:

- Menu Bar
- Navigation Panel
- Parameter Set Panel

Menu Bar

The Menu bar contains the following five sub-menus:

- **File**: provides an option to exit the DCW
- **Edit**: provides options to cut, copy and paste text in editable text boxes and cells
- **Level**: provides options that allow you to add, delete and rename parameter levels
- **Parameter**: provides options that allow you add or delete parameters from a parameter level
- **Help**: provides you information regarding the version of the DCW and access to online Help

Each of these features are described in detail in the sections that follow.

Figure 5: DCW Menu bar

Navigation Panel

The navigation panel, shown on the left in Figure 6, shows all of the data sources available in the system (the brackets show the data source version and optionally the data source format; e.g. for the EFU data source, the version is 5.0 and the format is FFF). When you expand a data source, a **Parameters** node appears for it along with any Level 1 parameters defined for the data source. In turn, each Level 1 node has a **Parameters** node and any Level 2 parameters defined for that parent level. As an example, Figure 6 shows the fully expanded EFU data source and all of its defined levels. The regions **North**, **South**, **East** and **West** are Level 1 parameters. The divisions

DIV1, DIV2, DIV3 and DIV4 are Level 2 parameters. The branches (i.e. **Branch3**) and products (i.e. **Mortgage**) are Level 3 and Level 4 parameters respectively.

Figure 6:   DCW Navigation panel with pre-configured data sources

✓ **Note** Existing data sources cannot be deleted and new data sources cannot be added through the DCW. Levels can only be added, deleted, or renamed. A data source's levels hierarchy can have a maximum depth of four (for example, in Figure 6, at the products level, **Mortgage** cannot be expanded further). There can be any number of parameter nodes defined at each of the levels (for example, in Figure 6, there could be any number of branches defined at Level 3). For more information on the values contained in these files, or to modify the default values, contact your Fair Isaac account representative.

Parameter Set Panel

The parameters for a data source are separated into logical parameter sets (or parameter types) by the DCW for editing convenience. When a **Parameter** node is selected in the Navigation panel, the Parameter Set panel is populated with a list of parameter types available for the governing data source (for example, in Figure 6, if any of the **Parameter** nodes under the EFU data source are selected, then **Inquiry** and **Analysis** appear in the Parameter Set panel). By selecting a parameter set, you can then edit the parameters in the set. For more information editing parameters, see “Managing Data Source Parameters” on page 97 and “Managing Level Parameters” on page 100.

- The **Parameter Sets** box allows you to select the parameter type, **Inquiry** or **Analysis**, to be edited in the Parameter Set panel. A data source parameter level is comprised of a subset of the entire list of parameters available from its data source.

- The left side of the Parameter Set panel is referred to as the Summary panel and it displays up to three tabs depending on the task being performed. The tabs on this panel are described in the sections that follow.
- The right side is referred to as the Parameters panel and can display one or more tabs with the parameters for a data source or parameter level(s). The tabs on this panel are described in the sections that follow.

  Note When both of these panels are active a splitter between them is activated to allow you to resize the panels as needed.

The tabs on the Summary and Parameters panels will change based on the Parameters that are selected and whether the **Show Hierarchy** check box is selected.

Show Hierarchy

Enabling **Show Hierarchy** lets you edit and view a summary of an entire parameter's hierarchy. For example in Figure 6, for data source EFU, when the parameters in the **Branch2** level are selected you can edit and view the parameters for the hierarchy that includes the levels **Branch2**, **Div2** and **North**. If **Show Hierarchy** is not enabled, then only the parameters for the **Branch2** level are available for editing and no hierarchy summary information is displayed. The sections that follow describe in detail the differences in the Parameter Set panel functionality when Show Hierarchy is disabled or enabled.

Show Hierarchy Off

If the **Show Hierachy** check box is not selected, the Parameter Set panel functions as described below:

- When **Parameters** at the data source level are selected, a single **Validation Errors** tab appears on the Summary panel. This tab displays errors that occur when invalid values are entered and saved in the Parameters panel. For more information on managing errors, see “Managing Errors” on page 104.
- When **Parameters** at any parameter level below the data source are selected, as shown in Figure 7, two tabs appear in the Summary panel, a tab with the data source name, version and format for the data source's default parameter values and the **Validation Errors** tab. On the Parameters panel, the name of the selected parameter level appears on a tab with the subset of the default parameters that have been added for this parameter level. For more information on adding parameters, see “Adding Parameters” on page 100.

Figure 7: Parameter Set panel showing the master list of inquiry parameters for data source EFU in the Summary panel and the inquiry parameters of the Branch1 parameter level in the Parameters panel, with the Show Hierarchy option off

Show Hierarchy On

If the **Show Hierarchy** check box is selected, all the parameter levels in the hierarchy associated with the selected Parameters are available for edit and the Parameter Set panel functions as described below:

- When **Parameters** at any parameter level below the data source are selected, as shown in Figure 8, three tabs appear in the Summary panel; the data source, **Merge,** and **Validation Errors**. The **Merge** tab shows a hierarchy summary with the full set of parameters, their values and the parameter levels from which they were merged. This provides a complete snapshot of the parameters that will be applied to transactions requesting the current levels hierarchy of parameters. On the Parameters panel, a tab appears for the selected parameter level with the subset of the default parameters that have been added for this parameter level. Additionally, a tab for each parameter level above the currently selected level appears and contains their respective sets of parameters.

Figure 8: Parameter Set panel showing the Merge tab in the Summary panel and the inquiry parameters of the EFU's North, Division 2, Branch 1 parameter level in the Parameters panel, with the Show Hierarchy option on

Command Buttons

The Parameter Set panel contains the following command buttons:

- **Save**: saves any changes made to the currently selected parameter set (Inquiry or Analysis) and clears the Summary and Parameters panels so another Parameter Set can be selected for edit. The **Save** button becomes enabled after an edit has been made.
- **Close**: exits the currently active Parameter Set panel and allows you to select another Parameter Set from the Parameter Sets list. The **Close** button becomes disabled after an edit has been made.
- **Cancel**: discards any changes to the current Parameter Set (Inquiry or Analysis) and clears the Parameter Set panels (Summary and Parameters) so another Parameter Set can be selected for edit.

Managing Parameter Levels

As mentioned previsously, each data source can have a hierarchy of up to four parameter levels. A full set of parameters (default parameters) is provided at each data source level. The data source default parameters are applied in every hierarchy for a selected data source. Settings for Level 1 override the defaults, settings in Level 2 override the settings for Level 1, and so on. Each data source can have an unlimited number of levels defined at each of the four allowable level depths.

Creating parameter levels, parameter sets, and editing the values of the parameters in each parameter set is a multi-step process including:

- Adding Parameter Levels below a Data Source
- Adding Parameter Levels below a Parameter Level
- Managing Data Source Parameters
- Managing Level Parameters

Adding Parameter Levels below a Data Source

Parameter levels are added, deleted and renamed using the **Level** menu on the Menu bar. If mulitple parameter levels are being used, the first parameter level must be added at the data source level. Level 1 parameters can be added to a data source as described in the following instructions.

To add a parameter level to a data source

1. From the Navigation panel, select the data source.
2. From the Menu bar, select **Level > Add**.

Figure 9: Adding a paremeter level at the data source

A **New Level** entry appears in the tree in the Navigation panel.

Figure 10: Enter a name for the new parameter level

3 In the **New Level** box, enter a name for the new parameter level (the name must be one or more characters and be unique for the level).

The new parameter level appears in the Navigation panel with an empty parameter set. For information on adding parameters to a parameter level, see "Managing Parameter Levels" on page 93.

4 Repeat steps 1 – 3 to add the Level 1 parameter levels for each data source.

Adding Parameter Levels below a Parameter Level

In a hierarchical implementation, lower levels in the parameter level hierarchy must be added at the appropriate parameter level. Use the instructions that follow to add additional levels below a parameter level. At each of the four levels, you can have an unlimited number of levels.

To add a parameter level below a parameter level

1 From the Navigation panel, select the parameter level.

2 From the Menu bar, select **Level > Add**.

A **New Level** entry appears in the tree in the Navigation panel.

3 In the **New Level** box, enter a name for the new parameter level.

The new parameter level appears in the Navigation panel with an empty parameter set. For information on adding parameters to a parameter level, see "Managing Parameter Values" on page 97.

4 Repeat steps 1 – 3 to add each subsequent level of parameter levels that are required.

Figure 11: EFU data source with four parameter level hierarchy

Deleting Parameter Levels

Use the instructions that follow to delete parameter levels.

To delete a parameter level

1 From the Navigation panel, select the parameter level.

2 Do one of the following:

- From the Menu bar, select **Level > Delete**.
- Press the Delete key on your keyboard.

The parameter level is deleted.

Note If there are levels below the one being deleted, a warning appears stating that you are deleting mulitple levels and prompts you to confirm the deletion.

Renaming Parameter Levels

Use the instructions that follow to rename a parameter level.

To rename a parameter level

1 From the Navigation panel, select the parameter level.

2 From the Menu bar, select **Level > Rename**.

The edit box around the parameter level name becomes active.

3 Enter the new name for the parameter (the name must be one or more characters and unique for the current level depth).

Managing Parameter Values

Each data source has a default set of parameters that is delivered with your system. The DCW is used to maintain these parameters and can also be used to define levels which allow the parameters to be assigned hierarchically during processing.

Managing Data Source Parameters

The values assigned to each of a data source's default parameters can only be modified, you cannot add additional default parameters with the DCW. Items can only be added to list parameters (for example, CraScoringModelCode, or ExistingMemberNumber).

Assigning Parameter Values to a Data Source Parameter

Use the instructions that follow to assign parameter values at the data source level.

To assign a parameter value to a data source parameter

1 In the Navigation panel, expand the tree of a data source whose parameters you want to edit.

2 Double-click **Parameters** below the data source.

Depending on the data source, **Inquiry** and/or **Analysis** parameter types appear in the **Parameters Sets** box.

Figure 12: EFU Parameter selection

3 Double-click on either **Inquiry** or **Analysis** to edit a parameter in that default parameter set.

The default parameter set for the data source appears in the Parameter panel.

Figure 13: EFU Data source level parameter inquiry parameters in the Parameters panel

4 Click in the **Value** field of the parameter you want to edit.

Note If a parameter has a plus sign in the column to the left of the parameter name, it is a list paramater or it is a family of parameters. A parameter family is always treated as a unit. Family members cannot be added/deleted separately. Click the plus sign to edit the values in the list or family items (a list parameter contains the word “list” in its name). For information on adding more lists, see “Adding a New List Items to a List Parameter” on page 99.

5 Enter a value for that parameter.

6 Repeat step 4 and step 5 for each parameter value that needs to be edited.

7 Do one of the following:

- Click **Save**. Changes are saved and the current Parameter Set is closed.
- Click **Cancel**. Changes are discarded and the selected Parameter Set is closed.

  Data source and parameter levels are made available for selection in the Navigation pane.

Adding a New List Items to a List Parameter

Use the instructions that follow to add a list to a list parameter either at the data source level or at a parameter level.

✓ **Note** Use these instructions when adding a parameter family (a predefined parameter structure) at either the data source level or a parameter level.

To add a new list item to a list parameter

1. In the Navigation panel, expand the tree of a data source or paramater level whose list parameters you want to edit.
2. Double-click on the **Parameters** item at the selected level.

   Depending on the data souce, **Inquiry** and/or **Analysis** parameter types appear in the **Parameters** Set box.
3. Double-click on either **Inquiry** or **Analysis** to edit a parameter in that default parameter set.
4. Do one of the following to add a list:
   - From the Menu bar, select **Parameter > Add > <*list name*> > <*list*>**.
   - If the list is in the Parameter panel:
     - Double-click in the column to the left of the list item, or
     - Right-click in the column to the left of the list item and select **Parameter > Add > <*list name*> > <*list*>**.
   - If the list is in the Summary panel (when editing a level below a data source):
     - Double-click in the column to the left of the list item.
     - Right-click in the column to the left of the list item and select **Parameter > Add > <*list name*> > <*list*>**.

✓ **Note** If the list is full, a dialog box appears to notify you that you cannot add any additional lists.

Figure 14: Adding a new parameter list

Note A list parameter must contain at least one item.

5 For each new list, click in the **Value** field of the list paramater and add a value.

6 When finished adding values to the list, do one of the following:

- Click **Save**. Changes are saved and the current Parameter Set is closed.
- Click **Cancel**. Changes are discarded and the selected Parameter Set is closed.

  Data source and parameter levels are made available for selection in the Navigation pane.

Managing Level Parameters

The parameters for a level are a subset of the data source's default parameters. The level's parameters override any corresponding parameters lower in the hierarchy. For example in Figure 14, any parameters for **Revolving** will override the corresponding parameters in **Branch4** , **Div2**, **North** and **EFU[5.0FFF]**.

Adding Parameters

Use the instructions that follow to assign parameters to a parameter level.

To assign parameters to a parameter level

1 In the Navigation pane, expand the data source list and then expand the parameter levels until **Parameters** under the level you which to edit is available.

2 Double-click **Parameters** associated with the desired parameter level.

  Depending on the data source, **Inquiry** and/or **Analysis** parameter types appear in the **Parameters Sets** box

3 Double-click on either **Inquiry** or **Analysis**.

  The Summary panel displays the data source's default parameters. The Parameters panel shows the parameters for the level. When a level is initally added, it does not have any parameters defined it its parameter set.

Figure 15: Data source EFU, parameter level West inquiry parameters

4 Do one of the following:

- From the Menu bar, select **Parameter > Add > <*parameter name*>**.
- In the Summary panel:
  - Double-click on a parameter from the data source's default parameters.
  - Right-click on a parameter from the data source’s default list and select and click **Add <parameter name>**

Note Only parameter names that have not previously been assigned to this level will be available for selection.

Figure 16: Selecting parameters from the default list for the parameter level

The parameter is added to the Parameter panel for the selected parameter level and the cursor is in the value column of the Parameter panel so that the parameter can be assigned a value.

Figure 17: Parameter added to West parameter level parameter set under the EFU data source

5 Assign a value to the parameter. For more information, see "Assigning Parameter Values" on page 103

6 Repeat step 4 and step 5 until all the required parameters are added to this parameter level.

7 Optionally, add new list items to list parameters or parameter families. For information, see "To add a new list item to a list parameter" on page 99.

Assigning Parameter Values

Use the instructions that follow to add or change parameter values for a parameter level parameter.

To assign a parameter value to a parameter level parameter set

1 In the Parameter panel, click in the **Value** field of the parameter you want to edit.

✓ **Note** If a parameter has a plus sign in the column to the left of the parameter name, it is a list paramater or it is a family of parameters. A parameter family is always treated as a unit. Family members cannot be added/deleted separately. Click the plus sign to edit the values in the list or family items (a list parameter contains the word "list" in its name).

2 Enter a value for that parameter.

3 Repeat step 1 and step 2 for each parameter value that needs to be edited.

4 When all parameter values have been added, do one of the following:

- Click **Save**. The changes are saved and the Parameter panel is closed.
- Click **Cancel**. Any changes are disregarded and the Parameter panel is closed.

  Data source and parameter levels are made available for selection in the Navigation pane.

Deleting Parameters

Single and family parameters and can only be deleted from a parameter level parameter set, they cannot be deleted from the default parameter set of a data source. Only lists from data source parameters can be deleted. The data source list must have at least one list, parameter level lists can be deleted in their entirety. Use the instructions that follow to delete parameters, parameter lists and parameter families.

To delete single or family parameters

1 In the Navigation pane, expand the data source list and then expand the parameter levels until **Parameters** under the level you which to edit is available.

2 Double-click **Parameters** associated with the desired parameter level.

Depending on the data source, **Inquiry** and/or **Analysis** parameter types appear in the **Parameters Sets** box

3 Double-click on either **Inquiry** or **Analysis**.

The Summary panel displays the data source's default parameters. The Parameters panel shows the parameters for the level

4 Do one of the following:

- From the Menu bar, select **Parameter > Delete > <*parameter name*>**.

- In the Parameter panel right-click in the column to the left of the item and select **Delete <*parmeter name*>**
- In the Parameter panel click in the column to the left of the item to select it and then press the Delete key.

To delete list parameters

1 In the Navigation pane, expand the data source list and then expand the parameter levels until **Parameters** under the level you which to edit is available.

2 Double-click **Parameters** associated with the desired parameter level.

Depending on the data source, **Inquiry** and/or **Analysis** parameter types appear in the **Parameters Sets** box

3 Double-click on either **Inquiry** or **Analysis**.

The Summary panel displays the data source's default parameters. The Parameters panel shows the parameters for the level.

4 Do one of the following:

- From the Menu bar, select **Parameter > Delete > <*parameter name list*> <*parameter name x*>**. Where x is the number of the list you want to delete.
- In the Parameter panel right-click in the column to the left of the item and select **Delete <*parmeter name list*> <*parmeter name x*>**. Where x is the number of the list you want to delete.
- In the Parameter panel click in the column to the left of the item to select it and then press the Delete key.

  Managing Errors

When the **Save** is clicked, a validation is performed on all of the parameter values in the Parameter panel. If any validation errors are found the following dialog box appears:

Figure 18: Validation error dialog box

Additionally, an error is displayed on the Summary panel on the Validation Errors tab. All errors must be corrected before the parameter level can be saved.

Figure 19: Validation errors on the Summary panel

To correct validation errors

1 In the Validation Errors tab, in the Summary panel, double-click on any line item denoted with an [!E] or an [E].

The cursor moves to the value field of the parameter that the error is addressing.

2 Enter a valid value in the field.

3 Repeat steps 1 and 2 until all errors have been corrected, and then click **Save**.

The changes are saved and the Parameter Set panel is closed and the Navigation panel is now active.

Database Guides

Some of the writing samples in this section are rough approximations of the actual documentation that I published. Several of my samples are publicly available, but in some cases, I had to remove illustrations, product names, company names, tables, and entire sections, so the copyright owner would grant me permission to use excerpts in my portfolio.

- Creating the Aging Database User
- Consortium Data Extraction Guide

Creating the Aging Database User

Every installation of IBS includes a collection of SQL scripts and programs that will allow an aging user to automate the process of removing old data from the <REMOVED> Transfer database schema. This process is referred to as aging. The first step to setting up the aging process is to create an aging user who has the permissions necessary to manage the aging process. This section provides the instructions for creating the aging user in Oracle.

Important The aging user and the aging schema must reside on the same Oracle server as the <REMOVED> Transfer database.

Requirements

The following requirements must be met prior to setting up a new aging user:

- The DBA has been granted the necessary privileges.

  For more information, see "FIC_AGING user".
- The <REMOVED> Transfer schema already exists.
- The DBA has read/write access to the aging directory.
- The DBA has SQL Plus access at the UNIX prompt.

Creating an Aging User

Note It is strongly recommended that the FIC_AGING user has a default table space that is separate and independent from the IBS schema's table space and other Oracle user table spaces. The separate FIC_AGING user tablespace will provide enhanced performance and backup / restore opportunities as the database grows. To optimize performance, it is recommended that the aging user and the <REMOVED> Transfer schema be placed in different table spaces.

To create an aging user

1. At the UNIX prompt, change directories by typing: `cd <REMOVED> Transfer/DB/Database/Oracle/Aging`
2. Open `create_aging_schema_options.sql`
3. Use the following command to replace the default aging user's table space name with the name of the new table space:

   ```
   DEFINE Users_tablespace=newTableSpaceName;
   ```
4. Set the path of the directory where the archive files will be stored:

   ```
   DEFINE ARCHIVE_PATH = '/the/new/directory/path/for/archiving/files/'
   ```
5. From the Unix prompt, run the shell script:

   ```
   create_aging_user.sh (DBAUSER DBAPASSWORD FAL102 FICUSER)
   ```

   Where:

   - DBAUSER is the Oracle DBA user name.

- DBAPASSWORD is the Oracle DBA password.
- FAL102 is the Oracle database SID or alias.
- FICFPUSER is the <REMOVED> Transfer schema owner name.

This script produces a log file named FIC_AGING- < <REMOVED> Transfer schema owner> - $DATE.log.

The SQL script then grants many privileges to the FIC_AGING user. For a detailed list of the privileges, see "FIC_AGING user".

Note If you run create_aging_user.sh a second time for the same aging user, a new log file will be created, and the aging user will be recreated.

Once the FIC_AGING user has been created, the script will then:

- Create private synonyms pointing to the IBS schema's database objects.
- Create the tables necessary for the aging process to function.
- Create the packages necessary for the aging process to work.
- Configure the default aging criteria by inserting aging data into the aging tables.
- Schedule the add partitions job to add table partitions to the IBS schema tables.
- Schedule the default aging process to run nightly.

Changing the Aging User's Table Space

If an aging user was not placed in a separate table space when originally created, use the following command to move the user to a different table space.

To change an aging user's table space

▶ At the SQL prompt, type: ALTER USER < FIC_AGING user name DEFAULT TABLESPACE "<FIC_AGING user tablespace name>";

Changing the Aging User's Default Password

The SQL script will create a database user named FIC_AGING_< <REMOVED> Transfer schema owner> with a default password of AGING1. It is assumed that the DBA will use the Alter user command after the FIC_AGING user is created to change the password, although this is not necessary for the proper functioning of the aging process.

Setting Up the Aging Process

The aging process consists of two functions: scheduling and activation. Scheduling controls when the aging configuration is evaluated; activation of the aging processes are controlled by settings in the aging configuration tables.

For example, to schedule an aging sweep to start at midnight, the process will call the aging.process_tables procedure. This procedure evaluates the contents of the aging_parameter table and looks for the tables scheduled to be processed at midnight (plus 18 minutes). These tables will have a runtime between 0 and 0.3 (i.e., 18 minutes) as defined in the AGING_PARAMETER table.

Once these tables are identified, the partitioned tables with simple date based aging criteria are aged. The process will then evaluate the tables in the aging_table_filter table and age all matching tables. The aging_table_filter table is where more complex aging parameters can be defined.

The aging process can be started at any time by running the aging.process_tables procedure again. Each time the aging.process_tables procedure is executed, it evaluates the time of day and then ages the tables that have a matching time. Therefore, if the procedure was run at 10 a.m., the procedure would only process the tables with a matching runtime between 10 and 10.3. None of the tables scheduled to run at midnight would be aged.

Setting Up an Aging Schedule

This section provides information on scheduling aging processes using the default Oracle scheduler. The aging schedule can be set so that aging occurs during off-peak hours when it will have the least amount of impact on system performance. Changing of the runtime can be done using the aging schedule as well.

✓ **Note** Using the Oracle schedule is optional. Scheduling can also be controlled using Cron jobs. If a a cron job is used, be sure to remove the <REMOVED> aging jobs from Oracle. A cron process would need to log into Oracle as the aging user and issue the following command:

```
exec Aging.Process_tables;
```

Using Default Parameters

When the database aging user was created, the aging process was automatically scheduled using the default parameters that trigger the aging job to run daily starting at midnight (0) on the following day. The job name is FIC_AGING<time>. The default setup also schedules a job to run at each unique run_time set up in the AGING_PARAMETER table.

✓ **Note** The schedul_aging_job process only needs to be run once. If the runtime of any tables has been changed, the exec AGING_CONFIG.APPLY_CHANGES script must be run to drop all existing FIC_AGING schedules and recreate them based on the new aging criteria.

Using Your Own Parameters

Use the following steps to schedule the aging process to begin at a specific time.

To schedule the aging process

1 At the UNIX prompt change directories by typing: cd <REMOVED> Transfer/DB/Database/Oracle/Aging

2 To set the start date and hour parameters (start date, hour, job name), issue this command:

```
Exec aging.schedul_aging_job(sysdate, 2, ‘JOB_NAME’);
```

Modifying the Aging Process

When the aging user was created, a default setup script for the aging process was automatically installed. This script can be modified to fit the needs of your implementation.

For more information on parameters that can be changed to control the aging process, see "Aging Packages".

Important It is strongly recommended that data associated with active fraud cases not be aged from the database.

Log Into SQL Plus

All modifications to the aging process are done using update statements that are run through SQL Plus.

To log into SQL Plus

▶ At the UNIX prompt type: Sqlplus FIC_AGING_FT1/aging1@FT102

Where:

- FIC_AGING_FT1 is the IBS aging user name.
- aging1 is the default password for the aging user.
- FT102 is the name of the database instance.

Backing Up the Aging Parameters

Before modifying the aging process, it is recommended that the aging parameter tables be backed up.

To backup the aging parameter tables

▶ Issue this command: Exec aging_config.backup_curent_aging_config('N');

Restoring the Aging Parameters

The following command will restore the aging parameters from a backup copy.

To restore aging parameters from backup

▶ Issue this command: Exec aging_config.restore_curent_aging_config;

Turning Off Aging for a Payment Table

The following command will turn off the aging process for the payment table.

To turn off the aging process

▶ Issue this command: Exec aging_config.age_table(table name,'N');

Turning Off Data Archiving

Data archiving creates an external Oracle archive file. This feature can be turned off with the following command.

To turn off data archiving

- Issue this command: Exec aging_config.change_archive(table name, ‘N’);

Adjusting the Timing on a Portioned Table

Use the following command to schedule a job to start at a specific time (e.g., 2:30 a.m.) on a portioned table.

To change the start time

- Issue this command: Exec aging_config.change_time (table name, 2, 30);

Aging Tables Using Complex Criteria

The aging of some tables requires complex criteria and a more sophisticated approach to the process. For the aging of these tables, stored custom procedures, SQL WHERE STATEMENTS, and SQL UPDATE statements can be used.

Using Stored Procedures

It is recommended that stored custom procedures not be altered; however, the timing and other criteria can be adjusted in the aging_parameter table as the following examples illustrate:

To add or change the column and /or value used to age data

- Issue this command: Exec aging_config.change_aging_criteria(‘FP_CASE’,’CAS_STA’,’2’,’PARTITION_DT’,’25’,’Y’);

  Where:

  - FP_CASE is the name of the table that is being configured.
  - CAS_STA is the name of the column containing the data to evaluate.
  - 2 is the column value that will trigger aging.
  - PARTITION_DT is the name of the date column used to evaluate the age of the data.
  - 25 is the number of days that the data will be retained in the table.
  - Y is the replace_where flag that causes existing complex aging criteria to be overwritten.

To change the number of days that data is retained in a table

- Issue this command: Exec aging_config.change_days_kept(‘FP_CASE’,2,’Y’);

  Where:

  - FP_CASE is the name of the table that is being configured.
  - 2 is the number of days that the data will be retained in the table.
  - Y is the replace_where flag that causes existing complex aging criteria to be overwritten.

Using SQL WHERE Statements

Most of the more complex criteria that you will encounter when setting up the aging process can be expressed using SQL WHERE statements.

The aging criteria for these tables can be defined in the aging_table_filter table. For example, the LOGGING table's aging criteria that was generated by the AGING_TABLE_FILTER_BIUR trigger, and the Insert statement with example values, would appear similar to the following:

```
INSERT INTO AGING_TABLE_FILTER(TABLE_NAME, COLUMN_NAME, DATA_VALUE,
DAYS_TO_KEEP, DATE_COLUMN_NAME, WHERE_SQL)
VALUES ( 'FP_CASE', 'CAS_STA', '2',  30,'CAS_CLS_DT', NULL);
COMMIT;
```

The DATA_TYPE and WHERE_SQL columns were filled in by the table's trigger. The resulting WHERE statement would appear similar to the following:

```
WHERE CAS_STA = 2 AND CAS_CLS_DT < trunc(sysdate) - 2
```

If the date was omitted from column name, the WHERE clause would appear similar to the following:

```
WHERE CAS_STA = 2
```

This causes all FP_CASE rows to be aged when the CAS_STA column has a value of 2.

Another aging criteria can be added on the same table or on the same column. For example, to age the fp_case table when the CAS_STA column has a value of 5, and it's 7 days old. The Insert statement with example values would appear similar to the following:

```
INSERT INTO AGING_TABLE_FILTER( TABLE_NAME, COLUMN_NAME, DATA_VALUE,
DAYS_TO_KEEP, DATE_COLUMN_NAME)
VALUES ( 'FP_CASE', 'CAS_STA', '5', 7, 'PARTITION_DT');
COMMIT;
```

If the column name had been omitted, the generated statement would appear similar to the following:

```
WHERE 'PARTITION_DT < trunc(sysdate) – 7
```

✓ **Note** The WHERE_SQL column must be null in order for the application to generate the SQL.

Adding Logic to SQL WHERE Statements

When a table requires more complex criteria for proper aging, WHERE statements similar to the following can be created with the SQL used to age the rules_fired table:

```
'WHERE T.PARTITION_DT < TRUNC(SYSDATE) - 30 AND NOT EXISTS ( SELECT 1 FROM
CASE_PAYMENT F WHERE F.TRANSACTION_ID = T.TRANSACTION_ID) AND NOT EXISTS (
SELECT 1 FROM CIS F WHERE F.TRANSACTION_ID = T.TRANSACTION_ID) AND NOT EXISTS (
SELECT 1 FROM LEDGER F WHERE F.TRANSACTION_ID = T.TRANSACTION_ID)');
```

To represent complex relationships, the aging process always appends a "t" to alias the name of a table (e.g., From RULES_FIRED T). In this case, the target table, which is the value in the table_name column, will be aged when the record is at least thirty days old and is no longer associated with a case in the fp_case table.

Using Update Statements

To modify the number of days to retain data, while keeping the default aging criteria intact, use update statements similar to the ones that follow.

The following update statements will change the retention criteria to ten days for all tables that currently have a thirty-day retention:

```
update AGING_PARAMETER
set DAYS_TO_KEEP = 10
where DAYS_TO_KEEP = 30;
commit;
update AGING_TABLE_FILTER
set DAYS_TO_KEEP = 10,
where_sql = replace(where_sql,’- 30’,’- 10’)
where DAYS_TO_KEEP = 30;
commit;
update AGING_PARTITION
set DROP_DT = to_date(substr(PARTITION_NAME,-4),’MMDD’) + 10
where table_name in (select table_name from AGING_PARAMETER where DAYS_TO_KEEP = 30);
commit;
```

Stop the Aging Process

The following command will completely stop the aging of all tables.

To stop all aging processes

▶ Issue this command: Exec aging_config.stop_all_aging;

Restart the Aging Process

The following command will restart the aging process.

To restart the aging process

▶ Issue this command: Exec aging_config.start_all_aging;

Consortium Data Extraction Guide

The Consortium Data Extraction (CDE) process is comprised of a collection of scripts and executable files that simplify the task of extracting relevant data from the daily transaction data that is captured by Falcon Transfer. The CDE process extracts, encrypts, packages, and compresses both the transaction data and the associated case data. Live transaction data, combined with the case data produced by case analysts, contains an enormous amount of valuable information, which helps <REMOVED> determine how well Falcon Transfer is performing in production.

Data Security

Because the transaction data that is being collected and sent to <REMOVED> contains sensitive customer information, precautions have been taken to ensure that the entire end-to-end process of collecting, managing, and transferring this data is secure.

The actual values contained in the more sensitive fields such as customer account numbers are not needed. In order to associate multiple transactions from a customer and to connect fraud cases created by analysts, the CDE process protects sensitive information by encrypting all incoming account numbers using an algorithm. This algorithm generates a unique number that is the same length as the original account number but conceals its content while providing Fair Isaac the information needed to improve future model performance.

Encryption

Sensitive data is automatically encrypted on the <REMOVED> Server by the CDE process as it enters the system. Sensitive data is never stored to disk in its unencrypted form. The encryption is done using a private key that is generated by the CDE process.

Once encrypted, the data stays encrypted throughout the analysis process and is destroyed immediately afterward. <REMOVED> can extrapolates the majority of information required for analysis by using the field sizes and their encrypted contents.

Encrypted Fields

As Falcon Transfer captures live transactions, the CDE process encrypts the following data fields prior to saving any of the data to temporary storage.

Fully encrypted data fields

- AGENCY ACCT WITH MEMB
- BANK CUST NUM
- BENEF CUST ACCT ADDR
- BENEF CUST ACCT NAME
- BENEF CUST ACCT NUM
- CUST ACCT NUM
- CUST ID NUM
- FIRST NAME
- LAST NAME

- REF INFO
- REGULATORY REPORTING
- REMITTANCE INFO
- ORIG CUST ACCT ADDR
- ORIG CUST ACCT NAME
- STREET LINE 1
- STREET LINE 2

Partially encrypted data fields

- DOB — Encrypt columns 5-8 (MMDD), leave columns 1-4 (CCYY) in clear text.
- TELEPHONE — Encrypt columns 6-16, leaves columns 1-5 in clear text.

The Consortium Data Extract Process

Both live transaction data and case creation data contribute to analyzing the performance of models. The following sections describe how the CDE process captures information from live transaction data and case data.

Capturing Live Transaction Data

The CDE process uses a bulk loading file to write live transaction data into the database in large batches rather than as individual transactions. The information is stored in a bulk-load file until a pre-determined limit (size or schedule) is reached. Once this limit is reached, the CDE process writes the data to the database. The bulk-loader is much more efficient than writing individual transactions and has a lessor impact on system performance.

✓ **Note** If CDE process is disabled, the bulkload file is deleted after the data is loaded into the database.

Whichever the trigger, size or schedule, once the write process is underway, profile data that we use for analysis purposes is appended to the bulkload data and then the entire contents is written to the database in one large batch, thereby lightening the burden on your system that would be caused by saving individual transactions to the database separately.

Live transaction data alone, however, is only part of the picture. To analyze how IBS is performing, we must examine your live transactions in light of the fraud cases that your analysts investigate. It's necessary to analyze both data sets together in order to see the complete picture of how well IBS is performing.

Case Data

The CDE process also collects the case data that is produced by your analysts who are investigating potential cases of fraud. In order to gather this information, the CDE process runs a nightly query against the database and creates an archive file for the previous day.

Once the nightly query is complete, the CDE process combines the archived live data with the results of the query of the case data. It combines this information with several other files that define the structure and the data and writes all of it to an archive file. Finally, the CDE process combines the current daily archive with subsequent archives. The full archive can then be compressed and prepared for transfer.

Configuring Consortium Data Extract

Consortium Data Extract is always installed on the <REMOVED> Server at the same time the product is installed. However, during your server installation, you were prompted to enable the CDE process. If the CDE process was enabled at that time, nightly queries can be scheduled.

Once the CDE process is configured and enabled, it will extract data by itself, and the database queries will occur during off-peak hours. If the CDE process was not configured during Server Installation, the setup process must be re-run. For more information, see the *<REMOVED> Server Installation Guide*.

✓ **Note** During installation, the default keyfile filename is set to cde_keyfile, which can be changed following installation. The file permission should be set to Read/Writable for the user and no one else. If the permission is not correct, the extraction process will fail and will be logged to the $PMAX_HOME/bin/logs/cde.log file. For more information on creating and managing keyfiles, see “Appendix F” of the *<REMOVED> Fraud Manager Server 5.2c System Reference*.

Configuring Nightly Queries

Nightly database queries are controlled by a cron job. It is recommended that this query be run during times of off-peak usage. This query also launches packaging and compression processes managed by the CDE process.

To schedule nightly queries

▶ In the $PMAX_HOME/util/pmax_env file make the following settings:

* set the ENABLE_CDE flag

* edit the ARCHIVE_DIR parameter.

Both of these settings are set by the setup.ksh script.

Sample Cron Entry

A sample cron job is provided in ${PMAX_HOME}/util/cde.cron. The following cron entry schedules the shell script (cde_nightly.sh) to run at 2:00 a.m. daily:

```
-----
0 2 * * * /bin/ksh "pmax_home/util/cde_nightly.sh >>pmax_home/bin/logs/cron.log 2>&1"
-------
```

Managing Disk Space

When the CDE process is enabled during the server installation, it immediately begins capturing live data when the system comes online. If the CDE process is enabled and you do not schedule the cron jobs to run nightly queries, a new CDE partial archive is created every day until there is no more disk space.

If the CDE process was installed and enabled, and did nothing else, the data would continue to be stored inside a new CDE partial archive daily. At the beginning of each day, the system would automatically create another file.

Release Notes

Some of the writing samples in this section are rough approximations of the actual documentation that I published. Several of my samples are publicly available, but in some cases, I had to remove illustrations, product names, company names, tables, and entire sections, so the copyright owner would grant me permission to use excerpts in my portfolio.

- <Removed> 6.2 Release Notes

<Removed> 6.2 Release Notes

Introduction

Welcome to <REMOVED>® <REMOVED> Suite Decision Accelerator 6.2. <REMOVED> Decision Accelerator enables you to manage complex decisioning strategies and automatically decision credit applications using advanced rules management technology.

What's New in this Release

The following sections detail changes introduced with <Removed> 6.2.

Enhancements

Many enhancements were made to <Removed> 6.2. Some highlights are:

- **Reports**—The following reports were added; Decision Flow Analysis, Pricing Decision Analysis, System Decision by Score Range, and Override Tracking.
- **Validation in SMW**—You can now validate configured items from any SMW Navigation pane view and save or print a validation report. You can validate a single item, an item type, or all items in a view and easily view each item's validation status.
- **Data Method Test Feature**—A new Data Method Test feature has been added to the Data Method Editor that allows you run test scenarios on multiple data sets directly from the SMW.
- **STF Batch Facility**—A new facility which allows you to automate strategy testing via command line.
- **Data Method Browse feature**—This new feature allows you to view dependencies.

See Also For more information, see the *Capstone Decision Accelerator Getting Started Guide*, *Capstone Decision Accelerator User's Guide*, *Capstone Intelligent Data Manager Administrator's Guide*, and *Capstone Intelligent Data Manager Developer's Guide*.

Predefined Product Strategies

Predefined product strategies can now be installed using an import file. For more information, see the <REMOVED>.

System Requirements

The following requirements apply to deployment and production environments.

Note For installation on Windows, some of the required software can be automatically installed during Decision Accelerator installation. For more information, see the <Removed>.

Web Service Deployment Using Tomcat (Windows)

- Windows Server 2003
- Apache Tomcat 5.0.28
- Java 2 SDK 1.4.2_14

Web Service Deployment Using Tomcat (UNIX)

- AIX 5.3 or Solaris 9
- Apache Tomcat 5.0.28
- Java 2 SDK 1.4.2_14
- Blaze Advisor 5.5.1.6 (December 22, 2004)

EJB and Web Service Deployment Using an Enterprise Application Server (UNIX)

- AIX 5.3 or Solaris 9
- IBM WebSphere Application Server 5.1
- Java 2 SDK 1.4.2_14
- Blaze Advisor 5.5.1.6 (December 22, 2004)

EJB and Web Service Deployment Using an Enterprise Application Server (Windows)

- Windows Server 2003
- IBM WebSphere Application Server 5.1
- Java 2 SDK 1.4.2_14

Data Storage Service Requirements

One of the following SQL databases:

- MySQL 4.1
- Oracle 9i or 10g (Data storage is supported, but not the SMW Report Viewer.)
- SQL Server 2000 or 2005 (Data storage is supported, but not the SMW Report Viewer.)

Workstation (Web Browser Host) Requirements

Regardless of the deployment method and platform you use, workstations accessing the SMW must meet the following requirements:

- Microsoft Internet Explorer 6.0
- Java 2 SDK 1.4.2_14

✓ **Note** Java 2 SDK and Internet Explorer are free programs that can be downloaded from the Internet. Instructions for downloading these programs are included in the *Capstone Intelligent Data Manager Installation Guide*.

Hardware Requirements

The minimum hardware requirements are:

- Development server: 1 GB of disk space on a 2 GHz Pentium, with a minimum of 2 GB RAM
- Deployment server: Minimum of 1 GB of disk space on a 2 GHz Pentium, with a minimum of 2 GB RAM
- UNIX servers: Recommend 2x1GHz or higher CPU, with 4GB RAM

Technical Notes

This section includes information on known issues that you may encounter when using the SMW or processing transactions.

Known Issues

The following are issues you may encounter when using Decision Accelerator 6.2.

Decision Accelerator Calls to Intelligent Data Manager Not Working

Due to recent changes in the Decision Accelerator Web service, calls from Decision Accelerator to Intelligent Data Manager (or any Web service that is not message-based or that uses namespaces for its parameters) do not work. Ignore all references to Intelligent Data Manager in the documentation.

Authentication Passwords for Administrators on Solaris

The authentication.xml file shipped with Decision Accelerator is not valid for Solaris. Authentication passwords for the administrator need to be regenerated before you run the ant genrma command. For a description of the steps that must be taken to ensure the installation is successful on Solaris, see “Creating Passwords” in the *Capstone Intelligent Data Manager Administrator’s Guide*.

Retaining Changes to Authentication.xml and Authorization.xml Files

Edited authentication.xml and authorization.xml files are overwritten whenever ant genrma or the automated Decision Accelerator installation is run. To avoid losing any changes you have made to these files, copy them from their current folder to another location, and then paste them back into their original location afterward. In Decision Accelerator 1.2 the files will be located within the repository folder, for example FairIsaac/EOS/DA/Data/BlazeRepository.sysid.

Genrma Error Message Says to Supply a User ID and Password

If you are doing a manual installation (UNIX only) and genrma fails during the Decision Accelerator installation process, look for an error message that refers to an application needing to supply a user ID and password. If that message is there, remove the repository configuration file (com_blazesoft_repository_config.cfg) from the repository’s root directory. Do not remove the file repVersionAndAuth.cfg. After removing the configuration file, run genrma again. This problem can be avoided by deleting the repository configuration file from the Decision Accelerator da12.exe file before unzipping the file.

If the repository is empty (which it most likely will be immediately after running genrma), signing into the SMW as guest1 or guest2 will cause the SMW to display an error message. Sign out from the SMW and sign in again as admin, then create some objects in the SMW. This should allow guest users to log in to the SMW.

Issue with Java 2 SDK Version

Decision Accelerator cannot run using a version of the Java 2 SDK higher than v1.4.2_14. If your system automatically prompts you to install a newer version, ignore this message. On Windows, you can prevent upgrade prompts by changing the setting in **Windows Control Panel > Java Plug** and clicking the **Update** tab.

Sporadic Exception Occurs After Editing Product Components

A sporadic exception may be logged to the SMW localhost_log when renaming or editing components.

Exception Occurs When Null is Passed

Each data method function should return null when a null numeric argument is passed to it; however, it may also return an exception. When a data method function returns an exception, the data method immediately terminates, and its result is whatever the result variable value was at the time the exception occurred (usually null, but not always). Processing continues, but it is very likely that additional exceptions will occur as a result of the data method returning an indeterminate value. To avoid this, include a conditional statement prior to the function call that confirms the numeric input(s) are not null.

Field Set to Null in Data Method Does Not Have Null Value in Output XML

If a data method attempts to assign null to a field that has a non-null value (for example, a numeric field with value 3), the field's value will be set to null internally, and null will be used as its value when it is referenced later in the data method or elsewhere, as expected. However, the value of the field in the output XML returned by Decision Accelerator will be its value prior to the null assignment (for example, 3).

TimestampFormat Returns Empty String Rather than Null

The data method TimestampFormat function should return null if a null argument is passed to it. Instead, it returns an empty string.

Issue Concerning Supported Timestamp and Duration Data Types

At this time, data object references and rule conditions do not support <, >, <=, and >= in the operator list for timestamp and duration type fields.

Issues with the Data Method Expression Editor

You may find that you are unable to copy or paste text to or from the Data Method Expression Editor. If this happens, it may be due to a setting in the Java Runtime Environment (JRE). If so, add a line containing "permission java.security.AllPermission;" to the java.policy file of the JRE being used by the SMW.

The data method editor will allow you to create a data method that includes a numeric constant value with a comma instead of a period as the decimal separator (for example, 1,3 instead of 1.3). The data method will be saved without error; however, a Blaze Advisor error will occur at run-time.

Currently the mappings screen allows mappings only to integer, real, and string fields, plus numeric or string data methods. For boolean field types, you must write a "wrapper" data method that generates a string, real, or integer, and then use that data method in the mapping. (For example, write a data method that generates a numeric 0 or 1 for boolean false or true, and then use that data method in the mapping.)

Omitting Leading Zeroes Causes Errors

In data methods and data object references (DORs), real constants entered without leading zeroes before a decimal point will produce errors. Be sure to always include leading zeroes to avoid this issue (for example, 0.6).

Decision Flow Loop Issue

When you create a decision flow, you can create loops from one step back to the previous step or to another step further back in the flow; however, there is no feature in the decision flow or the step itself that lets you control how many times the loop will execute, thus creating the possibility of easily creating an “infinite loop.” When using a loop, you must be sure to include a decision flow result that exits the loop, and take care to ensure that result value is generated at some point, to avoid an infinite loop.

Data Object Reference “Return Last” Option Returns All

The data object reference “Return Last” option does not work as intended. Rather than returning only the last item that matched the specified conditions, it returns all items that matched the conditions.

SMW Window is Frozen and SMW Dialog Box is Missing

You may find that if you are working in specific dialog box in the SMW and then change to another task in Windows (for example, Microsoft Outlook), the SMW dialog box you were previously working in seems to have disappeared and the main SMW window is frozen. To gain access to the hidden dialog box (and regain access to the main SMW window), press Alt+Tab (press and hold the Alt key while continually pressing the Tab key) until the Java “coffee cup” is highlighted. This will bring the hidden dialog box forward and the SMW window will no longer appear frozen.

Decision Responses and Message List Entries

If an error occurs after a decision has been registered, the response from Decision Accelerator may include both DecisionResponse and MessageList objects. You should always check MessageList objects to determine if the returned error invalidates the DecisionResponse object, as Decision Accelerator will not automatically make that determination.

Blaze Advisor System Errors

In order to execute your strategies, when it receives the first request after it starts, the Process Server will attempt to compile your strategies for execution within the Blaze Advisor rule engine, and will terminate without processing the request if any compilation error occurs, *even if the compilation error occurs for a product different from the one being requested*.

It is recommended that you thoroughly test all changes you make in the SMW in a test environment, to ensure the changes result in a clean compile, and the desired results are achieved.

Under certain circumstances, the Process Server will terminate processing and return a MessageList element whose description is "Blaze Advisor System Error." In the *Capstone Intelligent Data Manager Developer's Guide*, you are instructed to contact Fair Isaac Product Support when you receive this error; however there are some known errors that you may be able to resolve without contacting Fair Isaac, by reviewing information in the SMW, and making appropriate changes to eliminate the error. These errors include the following:

- If, after referencing a data method in a data method sequence, the data method is edited and its result type is changed, a compile error will occur. Try to avoid changing the method's return type. If you must change it, remove and re-add the method to edit the data method sequence to restore the link to the data method. This error is not detected by the validation feature.
- If a data method is used in a rule and its result returns a null value when run, you will receive error 10030 in the output XML:

  ```
  <Description>Invalid Data in Schema in <RulesetName>:<RuleName></
  Description><Resolution>Ensure that the data in the XML payload is
  valid and resubmit the request.</Resolution>
  ```

  In this case, there is not a problem with the input XML, and you can ignore the error unless the null data method result is unexpected.
- The data method editor does not perform name validation on inputs or locals. For example, names that begin with a numeric character may cause a Blaze Advisor compile error. This error is not detected by the validation feature.
- In the data method editor, the type of the duration argument in the DurationValue function is not validated. If an incorrect type is used, you will receive a Blaze Advisor compile error. Also, the prototype of this function in the Fields and Methods tab incorrectly shows the type of this input as numeric, when it should be a duration.
- If an input XML document containing a repeating attribute is submitted (for example, Application/Finance/Income/@SourceType, with one set to "Primary", and one set to "Self-employed"), you will receive messages in the return XML indicating exceptions have occurred in processing the input XML document, and the Process Server may need to be restarted. Be sure to verify your input XML doesn't include duplicate attribute values.

Edits are Lost After Navigating to Another SMW Screen

You may make changes in the SMW without being prompted to save. This can occur when you navigate from one screen to another after editing items such as decision flows and data method sequences without having first saved the changes. Whenever edits are made, be sure to save your work before leaving the current screen or your changes may be lost.

Report Has Expired Error in the Report Viewer

If the Report Viewer is left idle past the time-out, you may receive the following error: "Error. The report source has expired." To continue viewing reports you must sign out and then back into the SMW.

Add/Remove Programs Only Shows the Last Program Installed

The Add/Remove Programs screen in the Control Panel only lists the last Decision Accelerator program that was installed. To uninstall a program not listed, navigate to the appropriate directory and run the uninstaller from there.

- For Decision Accelerator navigate to C:\fairisaac\EOS_smwuninst and run DAuninstaller.exe
- For Process Server navigate to C:\fairisaac\EOS_psuninst and run PSuninstaller.exe
- For Data Storage Service navigate to C:\fairisaac\EOS_dssuninst and run DSSuninstaller.exe

Server Log Messages on Startup

The following Data Storage Service warning messages, which can be ignored, may appear in the Server Log upon startup.

- <yyyy-mm-dd> <hh:mm:ss> 33, 556 WARN [Thread-0] config.OptionConverter (OptionConverter.java:170)   - Could not find value for key jcs.default.cacheattributes
- <yyyy-mm-dd> <hh:mm:ss> 33,566 WARN [Thread-0] control.CompositeCacheConfigurator (CompositeCacheConfigurator.java:365)   - Could not instantiate ccAttr named 'jcs.default.cacheattributes', using defaults
- <yyyy-mm-dd> <hh:mm:ss> 33,576 WARN [Thread-0] config.OptionConverter (OptionConverter.java:170)   - Could not find value for key jcs.default.elementattributes
- <yyyy-mm-dd> <hh:mm:ss> 33,576 WARN [Thread-0] control.CompositeCacheConfigurator (CompositeCacheConfigurator.java:410)   - Could not instantiate eAttr named 'jcs.default.elementattributes', using defaults.

The following SMW warning messages, which can be ignored, may appear in the Tomcat Server Log upon the startup of Tomcat.

- WARN – Could not find value for key jcs.default.cacheattributes
- `WARN – Could not instantiate ccAttr named 'jcs.default.cacheattributes', using defaults`
- WARN – Could not find value for key jcs.default.elementattributes
- WARN – Could not instantiate eAttr named 'jcs.default.elementattributes', using defaults

DSS Admin Console Removed

The DDS Admin Console has been removed.

Unexpected Mouse Behavior in Strategy Management Workstation

When opening tree or table screens or controls, you may see a tool tip or highlight next to the mouse that says “Click to activate and use this control.” This is a behavior introduced by a recent Microsoft patch. Click once with the mouse to activate the control; you can then use the mouse to manipulate the control.

Use of Single Quotes in String Rule Conditions not Recommended

The use of single quotes or apostrophes (‘) in string rule conditions results in garbage characters appearing in the string when the server is restarted. To correct rule conditions with this problem, strip out single quotes with a data method and then do the check.

Testing Data Extractor Error

If a “Content is not allowed in prolog” error message appears when using the Testing Data Extractor, it can be ignored.

Normalize Transform May Not Return Elements in Correct Order

The order of elements returned in an output XML document may not match the order of elements in the master schema (the order documented in the data model spreadsheet and the “<REMOVED> Data Model” chapter of the *Capstone Intelligent Data Manager Developer’s Guide*.

Database Connection to DSS Not Available

mySQL connections are being held open after DSS closes them. This results in connections not being available for subsequent calls, and over a long period of heavy use could result in the database connection to DSS becoming unavailable. To prevent this, set the max_connections to a higher value than default (100) to reduce the probability of this issue impacting the reporting database

SRL Exception Occurs if a Data Method Returns Null to an Integer Mapping

A “Product category does not exist” error message is returned if a product selection tree uses an integer mapping that is mapped to a data method returning null. If this occurs, rewrite the data method in a way that ensures that null is never returned (do not return null directly, and do not let the data method throw an exception before a result is assigned).

Sign Out Before Saving Changes Can Create New Browser Window

If you sign out before saving your current changes, <REMOVED> will ask if you want to save before closing the SMW browser window. If you try to Cancel out of this window, the SMW will display in a new window, without the navigator pane. Close this window and click the Back button on the confirmation window, and the browser will re-display the correct SMW screen.

Exception Occurs in STF When Making Client Call to WebSphere

If you attempt to use the native EJB client connection (Invocation Target set to WebSphere EJB - [Local]), you will get an error message and the request will fail. Use the “Invoke Web Service - [Generic EJB]” target instead.

Corrupt Command Bar When Viewing Version History on WebSphere

When viewing a properties screen on WebSphere, if you select History from the drop-down list in the command bar, the history information is properly displayed, but the command bar becomes temporarily corrupted. Click on another item in the Navigator pane and the command bar returns to normal.

Messages in WebSphere Log File When Starting SMW on WebSphere

Under certain circumstances, when you start the SMW Web application in WebSphere, you will see messages in the WebSphere log file indicating certain cache files could not be deleted. You can safely ignore these messages:

```
[3/21/07 8:26:54:988 PST]  d22a8f7 AppUtils      W ADMA0078W: Unable to delete file: C:\Program
Files\WebSphere\AppServer\installedApps\serverhost\Strategy Management Workstation
sysid.ear\SMW.sysid.war\WEB-INF\classes\cache\all\AllCache.data
```

"User Authentication Failed" Message After Restarting WebSphere

When you restart the WebSphere application server, or if you start all three <REMOVED> applications simultaneously, under certain circumstances the process server does not start properly. If this happens, you will receive error messages from the process server if you submit requests using STF or any other calling application. The first time this occurs, the error message indicates user authentication failed, and for subsequent requests the error message indicates you attempted to access a server after it was shutdown. Restart the process server to clear up this error.

DSS SQL Database Setup Script Errors

DSS SQL Database setup scripts have table names in different cases that cause errors when run on UNIX. To remedy this, edit the base.sql script, drop the bad tables manually, and rerun all scripts.

Effective Dates Not Working in Decision Flows

The effective dates feature is not working for decision flows, and you should not use these fields.

Testing a Data Method that Uses an Internal Service Fails

You cannot use the Data Method Test feature to test data methods that call services. In this case, use the STF to test the data method.

Unhelpful Error Message for Data Methods that Take Inputs

The data method editor test feature does not currently support testing data methods which require inputs. If you try this, you will receive the following error message: "An error occurred while executing the test". Create another data method which calls the one with inputs, and test that one instead.

Unhelpful Error Message for Data Methods With Null Result

When the Data Method Test feature encounters an error during data method test execution, you will see the following error message: "An error occurred while executing the test". This error may be caused by a number of things, one of which is improper data input.

If your data method (or any data method that it uses), attempts to access a field for which no corresponding XML attribute or simple element is included in your Test XML Input data, the data method test feature will return this error.

Be sure your test XML includes all required fields before you execute the test.

Defective Product Strategy Link in the Category Tab

When you enter the Product Strategy page by clicking a link on the Category tab, the list of existing product strategies does not appear, allowing you to rename a product strategy the same as an existing one. If you rename the product strategy, the rename occurs in the repository, but references shown in the SMW are not updated. References will be updated the next time you restart the SMW. It is recommended that you rename strategies only from the Item Type view.

Errors in Data Methods With an Object as Input

You cannot call the Attach function to attach an object that is sent as an input argument. Instead, copy the input to a local variable and attach it.

UNIX DSS Users Must Edit Web Services to Specify Endpoint URL

UNIX users who have imported the pre-defined product strategies (pps_export.xml) must edit the ws_DSS_ProcessingHistory and ws_DSS_StrategyTestingFacility Web services to specify the proper endpoint URL for DSS, as these are set to default values for Windows/Tomcat deployments. Additionally, if you use the predefine product strategy's Global <REMOVED>® Score internal service, you must re-save the is_GlobalFI<REMOVED>core internal service, to properly set the properties for UNIX. You do not need to edit the file; just check it out, save it, and check it in. See the *Capstone Decision Accelerator User's Guide* for more information on how to edit Web services.

Data Method Test—Potential Memory Usage Issue

The data method test feature may consume too much memory over an extended testing period, resulting in an exception during data method testing. If this happens, check the Tomcat memory settings, increase them if possible, and restart the SMW Web application or Tomcat itself.

Lost Description, Threshold, Effective Dates for Rulesets and Rules

Descriptions, thresholds, and effective dates for rulesets and rules are lost if an item is not saved before another item is added. Be sure to save rulesets and rules before adding additional ones.

Error on SmallBusiness Category During Validation

After you import the pre-defined product strategies (pps_export.xml), if you validate the small business category, you will notice an errant validation error. You can safely ignore this error.

Product Strategy Tree Errors Not Appearing When Using Validation Feature

When you use the Validation feature to validate product selection trees, errors are not shown in the SMW. If you rename or remove a product that is referenced by a product selection tree, you must remember to update the product selection tree, otherwise you will receive a compile error when you attempt to execute the product.

Erroneous "OutputStream already obtained" Exception in Server Log for Report Viewer

If you have deployed the SMW in WebSphere, when you view a report, you may see the following exception in the WebSphere error log. The message is benign, and has no effect on the report, so it can be ignored:

```
[Servlet Error]-[OutputStream already obtained]: java.lang.IllegalStateException: OutputStream already obtained
```

Cannot Import Categories That Have No Default Product Selection Tree

If you are migrating a repository to Decision Accelerator 1.2.5, and the repository being migrated has one or more categories for which there is no product selection tree, you must create a dummy product selection tree for each of those categories before attempting to run the Export Utility on that repository. Otherwise, the import will fail.

Characteristic Labels in Score Model Ranked Reasons are Misspelled

When you add a score model with a model type of Ranked Reasons, the labels for integer 22, 24, 25, and 26 are misspelled. As these labels will be corrected in the future, it is best to avoid using the incorrect names so you will not have to reconfigure them later.

Distanced Reasons Option Not Working in Score Models

When the distanced reasons option is selected in score models, the results are the same as the ranked reasons option.

Unable to Deploy and Use a Second Repository Generated in WebSphere

When you follow the instructions in the *Capstone Intelligent Data Manager Administrator's Guide* for generating a second repository using genrma in WebSphere, the deployment fails. There is no work-around for this problem at this time, so you cannot generate a second repository when using WebSphere.

Decision Accelerator Translates Escaped Characters Incorrectly in <REMOVED>TA

Special characters that must be escaped in XML (&<>'") are sometimes handled incorrectly when they appear in attributes of the <REMOVED>TA data type. This may prevent running batch tests through Decision Accelerator in a non-production environment. If you encounter this problem, please contact Fair Isaac support for further details and patch availability.

Installing/Uninstalling Information

This section includes information on updating from version 1.1 to 6.2, as well as possible issues you may encounter during installation.

You have the option of installing the Predefined Product Strategies that come with Decision Accelerator1.2.5 by importing the strategies into your repository after you have installed Decision Accelerator For more information, see Appendix A in the *Capstone Intelligent Data Manager Installation Guide*.

Important If you are upgrading from version 1.0 or 1.0.1 to 1.2, contact your Fair Isaac representative for assistance.

Installation Issues

Installation instructions are located in the *Capstone Intelligent Data Manager Installation Guide*, which is located in the fairisaac\EOS\DA\docs folder.

Using Text Editors

If you are not using the automated installation program, when using a text editor to modify files during the post-installation process, it is recommended that you do not use Windows Notepad. If you do, the formatting may not be displayed correctly.

Builder Script to Run Blaze

The *Capstone Intelligent Data Manager Installation Guide* says to run Blaze Advisor and check its version, but Decision Accelerator does not provide a builder.sh script. See Blaze Advisor documentation for instructions on creating a builder.sh file.

Cannot Install Only the STF Using the Custom Install Option

To install the STF you must do a complete installation of Decision Accelerator. You cannot use the Custom installation option to install only the STF.

DataExtractor Port Number Not Set in STF.cfg File in WebSphere Automated Install

The DataExtractor port number is not set in STF.cfg for the WebSphere automated installation.You must manually edit the STF.cfg file to correct this.installation. For example, if WebSphere port is 9080, DataExtractor URL should typically be: URL="http://localhost:9080/dss/services/DSS"

Environment-Specific Issues During Automated Installation

The automated installation of Decision Accelerator and Process Server may fail due to your host machine environment. The following is a list of some factors that are known to impact the ability for a user to cleanly install Decision Accelerator or Process Server:

- Existence of a CLASSPATH environment variable at system or user-level. This can be especially problematic if the CLASSPATH references jar files, as there may be conflicts in your jar files and those required during and after the Decision Accelerator installation procedure. It is recommended that you remove any CLASSPATH environment variable settings before attempting the automated install.
- Java SDK or JRE installed in a folder whose pathname includes white space (for example, C:/Program Files/java). If this is the case on your host machine, we recommend you re-install the SDK or JRE in a folder whose pathname does not include white space, or remove the SDK or JRE and let the automated installation program install it for you in the default location.
- Alteration of the default Java Classpath setting in Tomcat. If you have altered the Tomcat Java Classpath setting, the automated installation may fail. This practice is not recommended by Apache, and may cause problems during or after the automated installation.
- It is recommended you reset Tomcat's Java Classpath to the default setting (typically C:\Program Files\Apache Software Foundation\Tomcat 5.0\bin\bootstrap.jar).
- Attempt to use an incompatible version of Tomcat.

Replace Existing Object Message During Install

If you attempt to re-install Decision Accelerator 1.2 the following message may appear: "Replace existing object data exists on this system and is newer than the object being installed. Do you want to replace this object?" If this happens, select "Yes to All," when prompted.

Migrating from Previous Decision Accelerator Releases to 1.2.5

Repository changes have been made with Decision Accelerator 1.2.5, which require conversion of the 1.1 or prior 1.2 configuration repository data in order to make it usable with version 1.2.5. The Decision Accelerator migration utilities export data from your existing Decision Accelerator repository, convert the data into appropriate format and then import it into the Decision Accelerator 6.2 configuration repository. Three utilities are included: Export Utility, Transform Utility, and Import Utility. Each utility has a command line interface and runs outside of Decision Accelerator components such as the SMW and the Process Server. During the export and import process, the utilities back up the configuration repository and update version information for all instance files.

The log for the migration utilities is captured in MyLog.log files located at C:\fairisaac\EOS\DA\utilities\Migration\bin, assuming the default location was used during Decision Accelerator installation. The various severity levels (in the reducing order of severity) supported for logging are:

- SEVERE
- WARNING
- INFO
- FINE
- FINER
- FINEST
- ALL

Important

- Be sure to perform a backup of your system before starting the migration process.
- You must complete the Decision Accelerator 1.2.5 software installation before starting the repository migration. For information on installing Decision Accelerator 1.2.5, see the *Capstone Intelligent Data Manager Installation Guide*.

In order to complete the repository migration from Decision Accelerator 1.1 to Decision Accelerator 6.2, the following must be done:

1 Ensure the tasks discussed in "Before Performing the Migration" are completed.

2 Run the appropriate Export Utility:

  a If your repository was running in Decision Accelerator 1.1, run the Decision Accelerator 1.1 Export Utility.

  b If you repository was running from a prior Decision Accelerator 1.2 release, run the Decision Accelerator 1.2 Export Utility.

3 If the repository you exported was from Decision Accelerator 1.1, run the Decision Accelerator 1.1 to 1.2 Transform Utility.

4 Run the Decision Accelerator 1.2 Import Utility.

5 Incorporate the object model from the imported repository into Decision Accelerator 1.2.5.

6 Generate the SRL.

7 Remap and rename components.

Before Performing the Migration

Take note of the following before migrating strategies from the 1.1 repository to the 1.2.5 repository:

- The Decision Accelerator 1.1 repository must be valid before performing the export; each of the utilities will only work with valid XML files.
- Only Working product versions in the Decision Accelerator 1.1 repository can be migrated to the 1.2.5 repository. Before performing the migration, use the SMW's Make Working command as necessary to ensure that all of the products you wish to migrate are writable.
- If the repository you are migrating contains categories that do not include at least one product selection tree, before starting the migration you must create a product selection tree for each category. The product selection tree does not need any specific content, it just must be created.

Decision Accelerator Export Utility

During the export process, the Decision Accelerator Export Utility reads the repository and extracts the data into an XML file.

To run the Export Utility

1 Navigate to the directory where the migration utilities are located. For example, assuming the default location was used during Decision Accelerator installation:

   For Windows: C:\fairisaac\EOS\DA\utilities\migration\bin

   For UNIX: /fairisaac/EOS/DA/utilities/migration/bin

2 Open a command prompt and run the appropriate export batch file or shell script for the repository you are exporting, as follows:

   - For a Decision Accelerator 1.1 repository, on Windows, execute runExportDA11.bat, and on UNIX, execute runExportDA11.sh.
   - For a Decision Accelerator 1.2 repository, on Windows execute runExportDA12.bat and on UNIX, execute runExportDA12.sh.

3 Provide the following parameters for the selected migration utility:

   - Repository Dir (for example, C:\fairisaac\EOS\DA\data)
   - Repository Name (for example, BlazeRepository)
   - Repository Version (for example, 1.1.0.68)
   - Destination Location + OutputXMLFileName (for example, C:\export11_output.xml)
   - Repository user ID (for example, admin)

- Repository password (for example, admin)

The Export Utility connects to the Blaze repository, reads the contents of its repository and writes them to the specified export file in the appropriate export file format.

If an error occurs while processing, the Export Utility displays the error and terminates processing. Errors are written to the MyLog.log file found in C:\fairisaac\EOS\DA\utilities\migration\bin.

Before proceeding with the Transform Utility, check the MyLog.log file to confirm that no errors occurred.

Decision Accelerator 1.1 to 1.2 Transform Utility (For 1.1 Repositories Only)

If you are migrating a repository from Decision Accelerator 1.1, before importing the repository you need to also execute the Transform Utility, which transforms the export file to the Decision Accelerator 1.2 export file format, and prepares it for import.

The Transform Utility is not used if you are migrating a repository from a prior Decision Accelerator 1.2 release.

To run the Transform Utility

1. Navigate to the directory where the runTransform.bat file (Windows) or runTransform.sh script (UNIX) is located. For example, assuming the default location was used during Decision Accelerator installation:

   For Windows: C:\fairisaac\EOS\DA\utilities\migration\bin

   For UNIX: /fairisaac/EOS/DA/utilities/migration/bin

2. Open a command prompt and run the runTransform.bat file (Windows) or runTransform.sh script (UNIX) with the following parameters:

   - Exported XML File Location (for example, C:\export11_output.xml). Refer to "To run the Export Utility" on page 135 for the name of the export file you specified during the import.
   - Transformed XML File Location (for example, C:\transform_output.xml)
   - Repository Version (for example, 1.2.0.0079)
   - Debug Level (for examples, see the severity levels listed starting on page 134)

   The Transform Utility reads in the Decision Accelerator data from the exported XML file location you specify and applies transformation rules to generate the Decision Accelerator 1.2 export file, in the Decision Accelerator 1.2 export file format.

   If an error occurs while processing, the Transform Utility displays the error and terminates processing. Errors are written to the MyLog.log file found in C:\fairisaac\EOS\DA\utilities\migration\bin.

   Before proceeding with the Import Utility, check the MyLog.log file to confirm that no errors occurred.

Decision Accelerator 1.2 Import Utility

The Import Utility imports a Decision Accelerator 1.2 export file (created either by the Decision Accelerator 1.2 Export Utility, or the Decision Accelerator 1.1-1.2 Transform Utility), into a Decision Accelerator 1.2 configuration repository. The Import Utility creates a backup of the repository. You can revert to the backed up repository at any point in time after import if you need to rerun the import.

Important The Import Utility should be run on a new/blank repository. For information on creating a new repository, see the "Creating Additional Configuration Repositories" in the *Capstone Intelligent Data Manager Administrator's Guide*. Running the Import Utility more than once with the same repository name and location will create duplicate strategies and strategy components. Do not run the import twice on the same repository. If you need to run the Import Utility again, you should first recover your blank repository from backup.

To run the Import Utility

1 Navigate to the directory where the runImportDA12.bat file (Windows) or the runImportDA12.sh script (UNIX), is located. For example, assuming the default location was used during Decision Accelerator installation:

For Windows: C:\fairisaac\EOS\DA\utilities\migration\bin

For UNIX: /fairisaac/EOS/DA/utilities/migration/bin

2 Open a command prompt and run the runImportDA12.bat file or the runImportDA12.sh script (UNIX) with the following parameters:

- Transformed XML File Location (for example, C:\transform12_output.xml)
- DA1.2.5 Repository Location (for example, C:\fairisaac\EOS\DA\data)
- DA1.2.5 Repository Name (for example, BlazeRepository.sysid)
- Debug Level (for examples, see the severity levels listed starting on page 134)
- Repository user ID (for example, admin)
- Repository password (for example, admin)

The Import Utility will read in the data from the transform12_output.xml file and import the data into the Decision Accelerator 1.2.5 configuration repository.

If an error occurs while processing, the Import Utility will display the error and terminate processing. Errors are written to the MyLog.log file found in C:\fairisaac\EOS\DA\utilities\migration\bin.

Before proceeding with the SRL generation, check the MyLog.log file to confirm that no errors occurred.

Incorporating Your Repository's Custom Object Model into Decision Accelerator 1.2.5

If you have modified the Decision Accelerator data model to incorporate elements and attributes other than those provided with Decision Accelerator, you must take additional steps to utilize your object model with Decision Accelerator 1.2.5. Failure to do so will result in failure of the SRL generation and will prevent you from being able to use Decision Accelerator 1.2.5 with this repository.

To incorporate your custom object model into Decision Accelerator 1.2.5

1. Review the "<REMOVED> Data Model Constraints" section found in the "Customizing the Data Model" chapter of the *Capstone Decision Accelerator Developer's Guide* to familiarize yourself with changes in Decision Accelerator data model constraints from Decision Accelerator 1.1 to 1.2.5.
2. Using an XML editor, such as XML Spy, add to your Decision Accelerator object model all of those mandatory Decision Accelerator 1.2.5 data model elements and attributes that are described in the developer's guide, and that were not required for your repository.
   - a If you are migrating a Decision Accelerator 1.1 repository, the Decision Accelerator 1.1 object model is in the file named <REMOVED>.SMWDisplayValues.xsd, located in the folder named C:\fairisaac\EOS\Common\data\schema (Windows) or <EOSPath>/fairisaac/EOS/Common/data/schema (UNIX) on the SMW server host where your Decision Accelerator 1.1 installation was located. Be sure to make a copy of the file for editing purposes.
   - b To assist you in this effort, Decision Accelerator includes a file <REMOVED>.DA.MinimalSchema.xsd, found in the folder named C:\fairisaac\EOS\Common\data\schema\<sysid> (Windows) or <EOSPath>/fairisaac/EOS/Common/data/schema/<sysid> (UNIX) on the SMW server host (where <sysid> is the system id of the repository). This file contains all the required elements and attributes.
3. Use the schema validation feature of your XML editor to ensure the file is valid and save it as <REMOVED>.SMWDisplayValues.xsd.
4. Follow the instructions provided in the *Capstone Decision Accelerator Developer's Guide*, in the section titled "Node-Level Changes in Data Model" in Chapter 11 "Customizing Data Model," to import your custom Decision Accelerator 1.2.5 data model into Decision Accelerator.

SRL Generation

After running the Import Utility, you must use the instructions that follow in order to generate SRL for the newly imported repository. Decision Accelerator must be operational before attempting this procedure.

To generate SRL for migrated strategies

1. Ensure that Decision Accelerator 1.2.5 is installed and operational.
2. Complete the repository migration detailed in the previous sections.
3. Launch the SMW. Be sure to use a role with administrative rights.

Note The SRL generation process uses the standard SMW logging mechanism.

To enable SMW logging, enter the following URL in the SMW browser:

http://localhost:<port number>/SMW.sysid/servlet/FlowServlet?Debug=true

where <port number> is the Tomcat port designated during installation.

For more information on SMW logging, see the *Capstone Decision Accelerator User's Guide*.

4 To generate the SRL for all the required components, enter the following URL in the SMW browser:

http://localhost:<port number>/SMW.sysid/servlet/FlowServlet?PostMigration=true

where <port number> is the Tomcat port designated during installation.

If an error occurs while processing, an error will display and terminate processing. Errors are written to the stdout.log file found in C:\Program Files\Apache Software Foundation\Tomcat 5.0\logs.

If no errors occur, migration to Decision Accelerator has been successfully completed.

Remapping and Renaming Decision Accelerator 1.1 Repository Components

When migrating a Decision Accelerator 1.1 repository, in most cases, the Import Utility creates a single analogous Decision Accelerator 1.2.5 component for each Decision Accelerator 1.1 component. For example one Decision Accelerator 1.2.5 ruleset is created for each Decision Accelerator 1.1 ruleset. The Decision Accelerator 1.2 Import Utility creates a single product strategy for each product, and that product strategy will be functionally equivalent to the Decision Accelerator 1.1 champion/challenger or standard strategy. In addition, for each Decision Accelerator 1.1 strategy the Import Utility creates a single Decision Accelerator 1.2.5 decision flow which is functionally equivalent to the Decision Accelerator1.1 strategy's decision flow.

See Also For more information on using Decision Accelerator 1.2.5 strategies and decision flows, see the *Capstone Decision Accelerator User's Guide*.

Some strategy components were renamed, and the structure of the components was modified, in the 1.2 release. Beginning with the 1.2 release, components that were previously defined at the product level are now defined at the system level, including:

- Data method sequences
- Decision flows
- Decision tables
- Decision trees
- Product strategies and decision flows
- Rulesets

Because these are no longer contained at the product level, in order to avoid any name conflicts after the Import Utility runs, components of these types are renamed by the migration Transform Utility so that they include the category and product name.

For example, if a ruleset named CraDataValidationPrimaryApp had been in the PreapprovedCard product, in the CreditCard category in the Decision Accelerator 1.1 release, after the migration, the ruleset would be named CraDataValidationPrimaryApp_CreditCard_PreapprovedCard. Decision Accelerator 1.1 data method sequences defined at the category level will be renamed to include the category name, and Decision Accelerator 1.1 data method sequences defined at the system level will not be renamed at all.

Since Decision Accelerator 1.2.5 allows multiple product selection trees to be defined for a specific category, whereas Decision Accelerator 1.1 allowed a single product selection tree per category, the migration utilities will create a Decision Accelerator 1.2.5 product selection tree under each category, and its name will be set to default_product_selection_tree_<category_name>.

See Also For more information on naming strategy components and using the SMW, see the *Capstone Decision Accelerator User's Guide*.

Migration Issues

Export Fails if New Category and Product Not Checked In

Exporting from Decision Accelerator 1.1 fails if a new category and product have not been checked in to the repository before running the Export Utility.

Problem with Hard-coded "sysid" Reference

There is a problem with the hard-coded "sysid" reference in migration\importDA\util\CommonDataHandler.java. If you try to import into a repository that is not named with the default name "BlazeRepository.sysid", it does not work. Always use the default name.

Decision Accelerator Product Documentation

The following sections list the documents available with Decision Accelerator and where to find them, as well as addendums for specific documents.

Documentation Location

After Decision Accelerator has been installed, you can find Decision Accelerator documentation in the C:\fairisaac\EOS\DA\docs folder. The Decision Accelerator documentation set consists of the following:

- *Capstone Intelligent Data Manager Installation Guide*
- *Capstone Decision Accelerator Getting Started Guide*
- *Capstone Intelligent Data Manager Administrator's Guide*
- *Capstone Intelligent Data Manager Developer's Guide*
- *Capstone Decision Accelerator User's Guide*

In addition, you can access the Capstone Decision Accelerator Strategy Management Workstation Help through the SMW interface.

✓ **Note** Information concerning pre-defined product strategies is included in the *Capstone Decision Accelerator User's Guide*. Note that specifics are given as example only; the product strategies delivered in your system may be different from those described in the Capstone Decision Accelerator documentation.

Support Information

Support is available to all clients who have purchased a Fair Isaac product and have an active support or maintenance contract. To expedite the resolution of your problem, please provide the following information:

- Your client ID or license number
- Your name, phone number, and email address or fax number
- The software product to which the problem pertains
- A description of the problem
- As much information as possible about your operating system, environment, and the conditions under which the problem occurred

For support on any Fair Isaac product, use one of the following methods to contact Fair Isaac.

Email

Put the product name in the Subject line of any email you send to <REMOVED> Product Support at support@fairisaac.com.

Telephone

- Asia-Pacific, Latin America, and the Caribbean: +1 (415) 446-6185.
- Europe, Middle-East, and Africa: +44 (0)870-420-3777.
- South Africa (toll free): 0800-996-153.
- UK (toll free): 0800-0152-153.
- US (toll free): 1 (877) 4FI-SUPP (1-877-434-7877).

Content Specification

Some of the writing samples in this section are rough approximations of the actual documentation that I published. Several of my samples are publicly available, but in some cases, I had to remove illustrations, product names, company names, tables, and entire sections, so the copyright owner would grant me permission to use excerpts in my portfolio.

- <REMOVED> 2.x Developer's Guide Content Specification
- <Removed> Administrator's Guide Content Specification

<REMOVED> 2.X
DEVELOPER'S GUIDE CONTENT SPECIFICATION

1 Revision History

Revision	Date	Comments
Draft 1	3/28/08	Created 2.0 Content Spec and started adding <REMOVED> 2.0 information.
Draft 1b	4/09/08	Applied Jane's edits.

2 Introduction

2.1 Purpose of document

The purpose of this Content Specification is to describe the <REMOVED> <REMOVED> Developer's Guide.

For general information about <REMOVED> <REMOVED> 2.0 documentation (product definition, marketing strategy) please see the <REMOVED> <REMOVED> 2.x Information Plan.

Document changes for subsequent 2.x releases will be appended to this plan.

Any revisions will be noted in the "Revision History" section above.

3 Product Description

3.1 Overview

<REMOVED> <REMOVED> (<REMOVED>) was originally developed to be the decisioning module in the <REMOVED> (EOS) suite of products. This is the last major release of <REMOVED> as part of the EOS suite. In the next major release, the product will be the decisioning module for<REMOVED> (EOP), part of <REMOVED>'s EDM suite of products. It is assumed that at that time, the name "<REMOVED> <REMOVED>" will be replaced.

The complete<REMOVED> (EOP) is to be comprised of the following:

- Decisioning Module (DM), which is <REMOVED> – Allows users to develop, test, deploy and execute business logic using rules and variables; available as end-user software.
- Data Acquisition Module (DAM), which is currently <REMOVED> <REMOVED> (<REMOVED>) – Provides connectivity to external data sources

such as CRAs and business bureaus. <REMOVED> 1.1.1 is currently in progress.

- Application Processor Module (APM) – Enables users to perform end to end application processing. APM 1.0 is currently in development.
- Analytics Module (AM) – Facilitates the development and deployment of custom analytic models and access to FI-developed <REMOVED>s models. This product has not yet been developed.

4 <REMOVED> Release 2.0

4.1 Release Overview

The primary purpose of the <REMOVED> 2.0 release is to upgrade <REMOVED> from Blaze Advisor 5.5 to Blaze Advisor 6.5. New functionality that was planned for this release was deferred in order to bring in the GA date.

4.2 Requirements Affecting Documentation Set

Reqs.Funct.	Description	Documentation Needed	Sections Affected
<REMOVED> support	<REMOVED> 1.0.1 and <REMOVED> 1.1 are supported. Supported Platforms Guide in SVN: <REMOVED>/DA/SupportedPlatforms/Delivered/<REMOVED>.DA.1.2.5.SupportedPlatformsDraft1.pdf	Reference the Supported Platforms Guide, which will show the versions of <REMOVED> that are supported.	Preface: Related Documentation Chapter 1: <REMOVED>

4.3 Bug Fixes Affecting Documentation Set

Note that the table below includes QC items listed in the Project Vision document as of 1/7/08. Details from Development are not expected until later in the project. This list is likely to change as we receive new information from Development and if new QCs are added to (or descoped from) the project. See the *<REMOVED> <REMOVED> Release 2.x Information plan*, March 24, 2008.

QC	Description	Documentation Needed	Sections Affected

2393	DG doc & ProcessServer.config have conflicting information on validateInput	The guide currently says that the value of the InvokerConfig/ DA@validateInput attribute should be set to yes. This assumes that schema validation should always be done; however that practice would significantly affect performance when in production. The documentation should recommend setting the default value to no with the caveat that you may want to set it to yes during implementation but change the value back to no when in production since this setting will affect performance.	Chapter 2: Data Exchange

4.4 Miscellaneous Changes Affecting Documentation Set

Reqs./Funct.	Description	Documentation Needed	Sections Affected
Product Version	Update product version number throughout documentation.	Change product version number to 2.0.	Cover page Copyright page: Version Number Preface: What's new in <REMOVED> 1.2 Glossary: Import Utility Glossary: Transform Utility

4.5 Audiences

The main audience for the Developer's Guide consists of developers who need to create a client application that interfaces with the Process Server. Clients' technical staff or a hired third party can perform this integration. Alternately, clients may opt to hire FI Consulting for this task. The guide explains the data model that underlies <REMOVED> and instructs developers on how to properly construct request messages and interpret response messages. It also provides information on extending the data model.

Alternate audiences for the guide include:

Business Rules Engineers (BREs): Before developing and implementing custom business policy using the SMW, BREs should become familiar with the DA data model.

Analytic Modelers (AMs): Before developing and implementing custom analytic models, AMs should become familiar with the DA data model.

Model Input Resources (MIRs): Before integrating client custom models with DA, MIRs can consult the guide for information that is not provided in the Blaze documentation.

GUI Engineers (GEs): Before modifying the SMW as requested by BREs or other stakeholders, GEs can consult the guide for information that is not provided in the Blaze documentation.

Operations Managers (OMs): OMs will need to understand the overall system architecture and the details of transaction processing.

QA/Testers: Testers will need to validate the implemented system against the information in the guide.

FI Consulting: Consulting may need to provide implementation services for Global <REMOVED> score, CrediTable models, and some custom model engagements. FI Consulting may also assist with other areas of implementation. Consulting may require information beyond what is revealed to clients.

System/network administrators (SNAs): SNAs will need to facilitate the deployment of the DA and/or <REMOVED> and configure all supporting hardware and network resources. Other tasks include monitoring the production execution environment and troubleshooting production issues.

5 Document Content

5.1 Book Versions

Only one version of the <REMOVED> DA Developer's Guide is necessary at this time

The external version of the guide is created by generating a FrameMaker book in its version folder. FrameMaker refers to files in the DevelopersGuides/External folder to generate the new book.

Product Version	Guide	Audience
DA 2.0	<REMOVED> <REMOVED> Developer's Guide	External clients of all sizes and geographies.

Editing of content for the guide takes place in the master files located in the DevelopersGuideMaster folder under the Shared Resources folder. The DA and <REMOVED> versions of the guide are created by generating FrameMaker books in the respective version folders. FrameMaker refers to files in the version folder and the DGMaster folder to generate the new book.

5.1.1 Conditional Text Tags

Tags used in the Developer's Guides:

Tag	Color	Applies To…
Comment	999999	Comments that should not be shown in any published book version. This tag has been used to temporarily hide information that we are not ready to show.
DA	CC3366	Text that applies only to DA.

GlobalFi<REMOVED>core	DkGreen	Text that only applies to Global <REMOVED> Scores. For now, we plan to show this text to all clients.
<REMOVED>	996600	Text that applies only to <REMOVED>.
Internal	660066	Comments that should only be shown in internal book versions (if any). This tag has been used to temporarily hide information that we are not ready to show.

At this time, there are no plans for additional conditional text tags.

5.1.2 Conditional Tag Use in Versions

The following table shows how the conditional text tags are applied to each version of the Developer's Guide.

Tag	Master	DA
Comment	On	Off
DA	On	On
GlobalFi<REMOVED>core	On	On
<REMOVED>	On	Off
Internal	On	Off

5.2 Publication Content

5.2.1 Front Matter Contents

Title page	Product Name Book Type Subtitle
Copyright page	Legal notices, Copyrights, Disclaimers Version Number, Template Number, Revised Date, Document Revision
Table of Contents	Three Levels of Headings
Preface	About <REMOVED> <REMOVED> What's New in <REMOVED> 2.0 Purpose of this Guide Conventions Used in this Guide Related Publications Contacting <REMOVED> Support Related Services Technical Publications About <REMOVED>

5.2.2 Body Contents

Chapter 1: Overview	<REMOVED> Products The <REMOVED> Product The Strategy Management Workstation The <REMOVED> Product Implementation Options The <REMOVED> Process Server <REMOVED> Data Storage Service Reporting Service The <REMOVED> Data Model Data Object References and Data Methods Deployment Options Communicating With External Systems Dataflow Options Subscriber Numbers Business IDs Joint Credit Report Requests ARF Input Parameters Response Data Transaction Resubmission Process Server Deployment Options Web Service Deployment <REMOVED>-Only Implementation <REMOVED>/<REMOVED> Implementation EJB Deployment <REMOVED>-Only Implementation <REMOVED>/<REMOVED> Implementation XML Processing Other Tasks for Interface Developers
Chapter 2: Data Exchange	Data Exchange Overview Web Service Client EJB Client XML Input and Output XSD Schema Files XML Processing Guidelines for Creating Valid XML User-Defined Fields Data Types Special Values Parse Errors

Chapter 3: The <REMOVED> Data Model	Data Model Elements
	Complex and Simple Elements
	Data Model Element Listing
	Data Model Spreadsheet
	Data Model Element XML Examples
	Analysis
	Applicant
	ApplicantAddress
	...
	Vehicle
	VehicleLease
	XpbRiskScoreReasonCode
	Data Source Parameter Levels
	UserDefinedField Objects
	DataMethodHistoryObjects
	<Removed Chapters 4 - 10>...
Chapter 11: Customizing the Data Model	Customization Overview
	Development Versus Production Environments
	Re-importing the Data Model
	Enumeration-Level Changes in Data Model
	Node-Level Changes in Data Model
	Transaction Output Changes
	<REMOVED> Data Types
	<REMOVED> Data Model Constraints
	Data Model/XML Naming Conventions
Appendix A: Troubleshooting	Error Handling
	<REMOVED> Error Handling
	<REMOVED> Dataflow Manager Error Handling
	Message Levels
	Message Summarization
	Error and Warning Message Lists
	<REMOVED> Messages
	Process Server Messages
	<REMOVED> Dataflow Manager Messages
	What To Do Before You Contact Product Support
	Gather Relevant Information
	Troubleshooting Checklist
	Contacting Product Support

5.2.3 Back Matter Contents

Glossary	Glossary of DA Terms Used in Developer's Guide
	adverse action
	aligned score
	alignment factor
	alignment multiplier
	<removed glossary terms>...
Index	XML Nodes
Subject Index	Topics

5.3 Version Contents

The book files for the various versions can reference chapter files from the DevelopersGuideMaster folder or from the same folder that holds the Book file for the version.

In the following table, M indicates that the file is located in DGMaster; B indicates that the file is located in the same folder as the version book; S indicates that the file is located in the Shared Resources folder.

Section	Location
Title page	B
Copyright page	S
Table of Contents	B
Preface	S
chapter: Overview	S
chapter: Data Exchange	S
chapter: The <REMOVED> Data Model	S
chapter: Consumer Credit Report Data	S
chapter: Business Credit Report and LOS Data	S
chapter: <REMOVED> Expansion Score Report Data	S
chapter: SBFE Data	S
chapter: Transaction Data and Decisioning	S
chapter: Processing XML Documents	S
chapter: Implementation Details	S
chapter: Customizing the Data Model	S
chapter: Global <REMOVED> Scores	S
appendix: Troubleshooting	S
Appendix: Parsed ARF	S
Glossary	S
Index of Topics	B

5.4 Graphics

The Developer's Guide contains the following graphics:

- System physical architecture diagrams in Overview.
- Schema (high-level view).
- Data Exchange diagrams.
- Data Source Parameter Levels in The <REMOVED> Data Model chapter
- Diagrams showing output results in the XML Processing chapter.
- Modification diagram in Customization chapter.
- Decision Flow Diagrams in Global <REMOVED> Scores Chapter

- Symbols are used throughout the guide in conjunction with the Tip, Note, See Also, Important, and Caution paragraph styles.

6 Document Delivery

6.1 Delivery Format and Location

The Developer's Guides are delivered as PDF, using the 8.5 x 11 FrameMaker template. The PDFs are to be delivered to clients through e-mail and will be available internally on the Product Documentation Team Services site under Product Documentation > Product Documentation > <REMOVED> > <REMOVED>.

6.2 File Identification

Developer's Guide documents are owned by Philip GeLinas and stored in SubVersion: <REMOVED> > DA > DeveloperGuides document repository.

Most figures and icons for the Developer's Guide are stored in the Figures subfolder of the DevelopersGuides > External > Figures folder. PDFs of each version are stored in DevelopersGuides > External > PDFs subfolder.

6.3 Change Documentation

Because the Developer's Guide documents are updated frequently, important changes must be communicated to clients and staff.

For a detailed listing of the changes to be made to each part of the Developer's Guide, please refer to DevelopersGuides > ChangeDocs > ChangeLog.<REMOVED>.DG.DA.doc. This document is for internal <REMOVED> use only.

For a listing of the nodes that have been changed since the last printing of the Developer's Guide, refer to the latest version of the DA change documentation: <REMOVED> > DA > DevelopersGuides > ChangeDocs > DocChanges.<REMOVED>.DG.DA.ccyymmdd.doc

7 Review Team

Reviews of the Developer's Guides should take place frequently to ensure accuracy of the information. Principal reviewers of the Developer's Guides are as follows:

Leonard <REMOVED> (Development) *	Rebecca <REMOVED> (Product Management) *
Keith <REMOVED> (Development)*	Vidya Anil <REMOVED> (Development)
Ruthraiah <REMOVED> (Development)	Prabh <REMOVED> (Development)
Nishant <REMOVED> (Development)	Brenda <REMOVED> (Quality Assurance)
Edward<REMOVED> (Development)	Mike <REMOVED> (Product Training)
Mike <REMOVED> (Quality Assurance) *	Yvan <REMOVED> (Consulting)
Deepthi <REMOVED> (Project Management)	Jim <REMOVED> (Product Training)
Ricardo <REMOVED> (Consulting)	Jim <REMOVED> (Product Support)
Linda<REMOVED> (Product Documentation)	Krista <REMOVED> (Product Documentation)

Phil <REMOVED>(Product Documentation)	

We require official sign-off on documentation from the reviewers with asterisks beside their names. These reviewers must review the documentation thoroughly and provide feedback before the guides can be released to clients.

8 Schedule Information

8.1 Developer's Guide Deliverables

Total Work Days	Inception Phase		Elaboration Phase		Construction Iteration 1 Updates		Construction Iteration 2 Updates		Construction Iteration 3 Updates		Transition Phase	
	Dys	Draft Date	Dys	Draft Date	Dys	Draft Date	Dys	Draft Date	Dys	Draft Date	Dys	Final Date
5	--	--	--	--	2	4/23/08	--	--	1	7/1/08	2	7/17/08

9 Dependencies

Completion of Developer's Guide documentation by scheduled deadlines depends on timely delivery of information from Development and QA staff:

- Review of the Developer's Guide Content Specification and Drafts by Product Management, Development, Quality Assurance, Support, Consulting, and Product Documentation.
- Updated SRSs, design documents, and other pertinent items from Development would be useful if contained in the QC records.

10 Open Issues

10.1.1 When will <REMOVED> (Process Server & DSS) support web service security authentication (QC #2308)?

11 Closed Issues

None.

12 References

1. <REMOVED>. "EOP 1.0 – <REMOVED> 2.0 Project Vision: Product and Feature Requirements." San Diego, CA. 2007.
2. <REMOVED>. "<REMOVED> 2.0 – Project Plan." San Diego, CA. 2008.

3. <REMOVED> "<REMOVED> DA 2.x Administrator's Guide Information Plan" San Rafael, CA 2008.

<Removed> Administrator's Guide Content Specification

1 ADDENDUM FOR <REMOVED> 1.1.1

1.1 OVERVIEW

The primary purpose of the 1.1.1 release is to add Canadian bureau data access that was provided only to Van City in the 1.1.0 release. Other enhancements include incorporation of consumer and Small Business SEARCH v5.3 DLLs, and administrative interface changes for setting inquiry and analysis parameters based on assigned organizational levels (data source level parameters).

An interim build referred to as the 1.1.0 release has been sent to one client, Van City. The final release available to all clients will be referred to as version 1.1.1.

1.2 ENHANCEMENTS AFFECTING THE ADMINISTRATOR'S GUIDE

Description	Documentation Needed	Sections Affected

Access to Canadian CRAs Equifax Canada and TransUnion Canada. Includes: ² TU Canada 4.0 ² Equifax Canada 5.0 ² Pre-developed inquiry construction capabilities for Canadian consumer CRAs ² CBD print image ² CBC print image ² ARF input dataflow option ² Serial and parallel dataflow input ² Alternate bureau routing (only within Canadian CRAs) ² NARF output ² Stand-alone products: Numeric Inquiry for CB Report (US versions already available) ² Option to request add-on products available in SEARCH 5.3 (subscriber directory, security checks/services)	Add Canadian CRAs wherever the interfaces are listed.	Preface: Introduction Preface: About <REMOVED> <REMOVED> Chapter 1: <REMOVED>s Chapter 1: Using <REMOVED> with <REMOVED> Chapter 1: Using <REMOVED> Only Chapter 1: Web Service Environment Setup Chapter 1: EJB Environment Setup Chapter 2: (intro paragraph) Chapter 2: Setting up Subscriber Numbers Chapter 2: Modifying Subscriber Numbers Chapter 3: Managing Data Source Connection Settings Chapter 3: Request Manager Service Chapter 3: Starting or Stopping a Connection Request Service Chapter 3: Managing Connection Resources Chapter 3: Data Source Line Status Chapter 3: Event Viewer Chapter 3: Glossary: cra
Other SEARCH migration support (consumer Canadian and US, business US). Includes adding CBC (common consumer print image) to output options for all consumer US and Canadian CRAs	List CBC with output options	Chapter 1: Connection Manager

Ability to set parameters/preferences by administration level. Includes: ² Up to four user-defined admin levels desired (i.e., system id, business unit, region, product) ² Expansion of the existing Windows app (DCW); levels implemented as directory structure ² Allow maintenance of parameters/preferences by administration level ² Set analysis parameters, inquiry parameters, subscriber number, subscriber identification data	Add topic: Defining Subscriber Identification Data Add topic: Modifying Subscriber Identification Data Add topic: Setting Up Administration Levels Add topic: Setting Parameters by User-defined Administration Level System Id Business Unit Region Product Add topic: Window Application Levels Implemented as Directory Structure	Chapter 2: Dataflow Configuration Workstation
Add Joint Spousal inquiry option (control of output options lie on first prin)	Add joint spousal inquiry option	TBD

1.3 QCs affecting the Administrator's Guide

There are no QCs affecting the 1.1.1 version of the Administrator's Guide.

1.4 Miscellaneous Affecting Documentation Set

Description	Documentation Needed	Sections Affected
Add 1.1.0 addendum content.	Include information from the 1.1.0 addendum that was delivered to Van City.	Chapter 3: Managing Data Sour Connection Settings
Change reference to <REMOVED> and DA “modules” to “products” per Lorrie, 3/23.	Replace the word module with product. Replace the word modules with products.	Throughout Administrator's Gui
Update to latest department FrameMaker template (v5.2).	Convert from template LG5.0 to 5.2. Includes: ² Importing paragraph and character formats from 5.2 cust file to 5.0 cust file (then applying these formats to whole book). ² Updating Preface section where boilerplate text has changed.	Change version to 1.1.1.
Add registered trademark symbol (®) after every occurrence of “<REMOVED>”.		Throughout Administrator's Gui

1.5 Publication Content

Below is the outline of the <REMOVED> Administrator’s Guide showing changes for the 1.0.1 release. New sections are shown in *red italics*, changes to heading title names are shown in *green italics*.

1.5.1 Front Matter Contents

Title page	Product name, FI product line, book type, release number
Copyright page	Legal notices, copyrights, disclaimers, template number, revised date, document revision letter
Table of Contents	

Preface	About <REMOVED> <REMOVED>
	Conventions Used in this Guide
	Related Publications
	Contacting <REMOVED>
	Contacting <REMOVED>
	    Support
	        Email
	        Telephone
	        Internet
	    Related Services
	        Strategy Consulting
	        Conferences and Seminars
	    Technical Publications
	About <REMOVED>

1.5.2 Body Contents

Chapter 1: <REMOVED> <REMOVED> Overview	<REMOVED> Overview
	<REMOVED>
	<REMOVED>
	Using <REMOVED> with <REMOVED>
	<REMOVED> Process Server
	<REMOVED> Components
	Dataflow Manager
	Credit Reporting Agency Serial Dataflow
	Credit Reporting Agency Parallel Dataflow
	Credit Reporting Agency ARF Input Dataflow
	<REMOVED>® Expansion Score Dataflow
	Business Bureau Dataflow
	Business Bureau ARF Input Dataflow
	Equifax SBFE Dataflow
	Connection Manager
	Dataflow Configuration Workstation
	Connection Configuration Workstation
	Repositories
	<REMOVED> Workflow
	Deployment Options
	Using <REMOVED> Only
	Using <REMOVED> with other <REMOVED> *Products*
	Web Service Environment Setup
	EJB Environment Setup
Chapter 2: Using the Dataflow Configuration Workstation (print only)	Inquiry and Analysis Parameters
	Setting Parameters
	Setting up Analysis Parameters
	Setting up Inquiry Parameters
	Setting Up Subscriber Numbers
	Modifying Subscriber Numbers
	Defining Subscriber Identification Data
	Modifying Subscriber Identification Data
	Setting Up Administration Levels
	Setting Parameters by User-defined Administration Level
	System Id
	Business Unit
	Region
	Product
	Window Application Levels Implemented as Directory Structure
	Performance Reporting
	Running Reports

Chapter 3: Using the Connection Configuration Workstation	Launching the Connection Configuration Workstation
	Touring the Workstation
	System Security
	  SQL Server Security
	Registering Your System
	Using the CCW
	  Administering Data Sources
	    Managing Data Sources
	    Managing Data Source Connection Settings
	    Using a Proxy Server
	    Scheduling Data Source Outages
	Administering CCW Services
	  Request Manager Service
	  Data Source Request Service
	    Modifying Data Source Request Services
	    Starting or Stopping Data Source Request Services
	  Connection Request Service
	    Modifying a Connection Request Service
	    Starting or Stopping a Connection Request Service
	  Managing Connection Resources
	    Modifying Connection Resources
	  Managing Connection Resource Objects
	    Adding a Connection Resource Object
	    Modifying Connection Resource Objects
	  Monitoring CCW Events
	    Data Source Line Status
	    Event Viewer
	    Communications Trace

Appendix A Managing Repositories	Administering the Message Database Utilizing the Event Log
Appendix B: Troubleshooting	Types of Error Messages Using the Dataflow Manager Log File What To Do Before You Contact Product Support Gathering Relevant Information Troubleshooting Checklist

1.5.3 Back Matter Contents

Glossary	
Index	

1.6 ADMINSTRATOR'S GUIDE DELIVERABLES

Document	Draft1	Draft 2	Final
Content Spec	3/18/08	--	5/21/08
Administrator's Guide	4/18/08	5/1/08	05/12/08

1.7 REVIEWERS

Lorrie Hoopes (Product Management) *	Ganyue Hu (Project Management)
Leonard Look (Development) *	Joseph Nguyen (Development) *
Homi (Development)	Lin (Development)
Scott (Quality Assurance) *	Jedd (Quality Assurance)
Raju (Quality Assurance)	Robert (Product Support)
Jim (Product Education)	Ricardo (Consulting)
Jane (Product Documentation)	Mike (Product Education)
Krista (Product Documentation)	Phil GeLinas (Product Documentation)

We will require official "sign-off" on documentation from the reviewers with asterisks beside their names.

1.8 ISSUES

1.8.1 Open Issues

1.8.2 Closed Issues

1.9 REFERENCES

1. <REMOVED>. "<REMOVED> 1.1.1 PM Requirements" San Rafael, CA. 2007
2. <REMOVED>. "<REMOVED> 1.1.1 DevQA Estimates." San Rafael, CA 2007.
3. <REMOVED>. Quality Center Defect IDs: "134", "137", and "138".
4. <REMOVED>. "<REMOVED> – <REMOVED> Release 1.x Information Plan" San Rafael, CA 2008.

Use Cases

This section contains use cases I wrote for a project I worked on several years ago.

Download Purchased Track

Description

Permanent Download guides a customer through the process of purchasing and downloading an audio track.

Author

Phil GeLinas

Goals

The customer wants to purchase and download an audio track.

Pre Conditions

The customer is an existing customer and the system has customer account information for the customer.

The identity of the customer and the customer's account has been verified.

Items are currently being selected for addition to an order.

Post Conditions

Products approved for purchase by the customer are added to the order being established in Fullfill Products.

Parent Use Cases

None.

Extended Use Cases

None.

Included Use Cases

Basic Flow of Events

1 The customer searches the music catalog.

2 The system displays an empty purchase list with a zero total.

3 The customer selects an item for purchase.

4 The system updates the purchase list with a new item entry and the following information:

- TrackId
- TrackName
- Item Cost

5 The system calculates and displays the new purchase total.

6 The customer requests to purchase the selected item.

7 The system displays the following payment options:

- Major Credit Card
- Debit Card
- Paypal

8 The customer selects a payment option. The system obtains payment information.

9 The system computes the order subtotal, taxes, and order total.

10 The system displays the following information

- TrackId
- TrackName
- Item Cost
- Payment option selected
- Order subtotal
- Tax
- Order total

11 The customer commits to the purchase

12 The system records the purchase.

13 The system authorizes fulfillment.

14 The use case ends.

Transfer to a Portable Device

Description

Device Registration guides a subscriber through the process of transferring a downloaded track to a portable device.

Author

Phil GeLinas

Goals

The subscriber wants to transfer a downloaded track to a portable device.

Pre Conditions

The customer is an active subscriber with the service.

The identity of the customer and the customer's account has been verified.

The customer's subscription level includes the right to transfer downloads to a portable device.

Post Conditions

The subscriber's subscription download is successfully transferred to a portable device.

Parent Use Cases

Media Transfer

Extended Use Cases

Multiple tracks and albums.

Included Use Cases

Basic Flow of Events

A. Device Registration

1 The subscriber attempts to transfer a downloaded track to a portable device.

2 The system requests the serial number of the device to verify that the device is active for transfer.

3 The subscriber provides the serial number from a portable device.

    [Alternate: Invalid Serial Number (B)]

    [Alternate: Too many active devices (C)]

4 The system verifies that the portable device is active for transfer.

5 The system transfers the file to the portable device.

6 The use case ends.

Alternate Flow of Events

B. Invalid Serial Number

1 The subscriber provides an invalid serial number.

2 The system displays a message requesting permission from the subscriber to add a new portable device to the account.

3 The subscriber authorizes the new portable device.

[Alternate: Subscriber Declines to Authorize (F)]

4 The system displays a message that confirms the new portable device has been added successfully.

5 The use case resumes at A.5.

C. Too Many Active Devices

1 The system determines that adding a new device would exceed the maximum number of devices allowed by the subscription.

2 The system determines if the subscriber has exceeded the maximum number of deactivations for the month.

[Alternate: The subscriber exceeds the maximum number of deactivations (E)]

3 The system prompts the subscriber to deactivate one of the three active devices already in the system (identified by a friendly name).

4 The subscriber deactivates an active device.

[Alternate: Subscribes declines to deactivate a device (F)]

5 The system activates the new device.

6 The use case resumes at A.5.

E. Device Not Active for Transfer

1 The system cancels the transfer request.

2 The use case ends.

F. Subscriber Declines to deactivate device

3 The system cancels the transfer request.

4 The use case ends.

5 The subscriber provides a serial number number for a device that is not active for transfe

1 If the device is NOT ACTIVE for transfer, and your activation process is transparent, device is activated for user. Transfer begins. USE CASE ends.

2 If the device is NOT ACTIVE for transfer, and your activation process requires user participation, the client application requests the user's permission to activate the device. Client also requests that user provide a friendly name for the device.

3 If the user declines permission to activate device, the transfer stops. USE CASE ends.

4 If the user agrees to activate device, the client application sends a request to us to activate this device

5 If the activation succeeds, the client application finishes transfer. USE CASE Ends.

6 If activation fails because user has maximum allowable (N) active devices, the client application prompts user to deactivate one of the currently active devices (identified by a friendly name)

7 If the user selects a device to deactivate, the client sends a request to us to deactivate the device.

8 If deactivation fails because the user has reached the deactivation limit for that month, user gets an error message, and the process stops.

9 If the deactivation succeeds, the client application will send an activate request to us for the new device. Client completes transfer. USE CASE ends.

10 If the user does not choose a device to deactivate, the transfer stops. USE CASE ends.

License Renewal On Portable Devices

Description

Permanent Download guides a customer through the process of purchasing and downloading an audio track.

Author

Phil GeLinas

Goals

The customer wants to purchase and download an audio track.

Pre Conditions

The customer is an existing customer and the system has customer account information for the customer.

The identity of the customer and the customer's account has been verified.

Items are currently being selected for addition to an order.

Post Conditions

Products approved for purchase by the customer are added to the order being established in Fullfill Products.

Parent Use Cases

None.

Extended Use Cases

None.

Included Use Cases

Basic Flow of Events

1 The subscriber requests to download a set of tracks to a PC registered in the system.

2 The system sends one leaf license with each track.

3 The system also sends one root license that will expire after 33 days.

4 The subscriber transfers the downloaded tracks (and licenses) from his/her PC to a portable device.

5 The subscriber plays the tracks on the portable device for 30 days.

6 After 30 days, the subscriber logs into the system from a PC.

7 The system automatically sends a new root license to the subscriber's PC, which is valid for another 33 days.

8 After 30 days, the subscriber transfers new tracks to the (same) portable device, or the subscriber requests to synchronize the device.

[Alternate: Expired Root License On Portable Device (B)]

9 The system copies the new root license (with a higher priority than the old one) to the portable device.

10 The new root license with the higher priority overrides the old one, so the subscriber continues to play the tracks another 33 days without an interruption of service.

11 The use case ends.

B. Expired Root License On Portable Device

1 The subscriber attempts to play tracks on a portable device with a root license that is 34 days old.

2 The system displays a message that directs the subscriber to connect to a PC to synchonize licenses.

3 The subscriber synchonizes the portable device.

[Alternate: Expired Root License On PC (C)]

4 The use case resumes at A.10.

B. Expired Root License On PC

1 The subscriber attempts to synchonize iicenses on a portable device with a PC that has a root license that is 34 days old.

2 The system displays a message that directs the subscriber to log into the system to renew content licenses.

3 The user logs into the the system.

4 The system automatically transfers a new root license to the subscriber's PC.

5 The system displays a message requesting to synchonize the portable device.

6 The subscriber chooses to synchonize the portable device.

7 The system transfers a new root license to the portable device.

8 The subscriber plays tracks on the portable device for another 33 days without interruption of service.

9 The use case ends.

Web Writing

This section contains procedural documentation that I published online while at IBM Technical Studio.

- Set Up Virtual Private Networking On Your AS/400
- Set Up Your IBM HTTP Server on AS/400 to Serve the Internet

Set Up Virtual Private Networking On Your AS/400

In this topic, you'll set up a Virtual Private Network (VPN) that uses IP Security (IPSec) protocol and Layer 2 Tunneling Protocol (L2TP) to protect and conceal sensitive information from hackers and thieves who prowl the Internet.

Note Before you start, Check System Requirements for this topic.

You'll learn to:

1 Set Retain Server Security Data Value to 1

2 Set Up AS/400 IPSec Policy

3 Configure your L2TP terminator <Removed>

4 Configure AS/400 packet filtering <Removed>

5 Configure Windows 2000 for a VPN connection <Removed>

6 Start your AS/400 VPN server <Removed>

7 Start a VPN session <Removed>

Check System Requirements

Here's a list of requirements necessary to set up this VPN.

- OS/400 requirements include:
- OS/400 V4R5 (5769-SS1);
- Crypto Service provider (5769-AC2 or 5769-AC3);
- Digital Certificate Manager (5769-SS1 option 34);
- V4R5 Client Access Express (5769 XE1 with Operations Navigator and the latest service pack).

Windows 2000 requirements include:

- Windows 2000 Professional edition;
- Windows 2000 High Encryption Floppy Disk (for 3DES).

Once your system meets these requirements, you'll be ready to set up AS/400 IPSec policy.

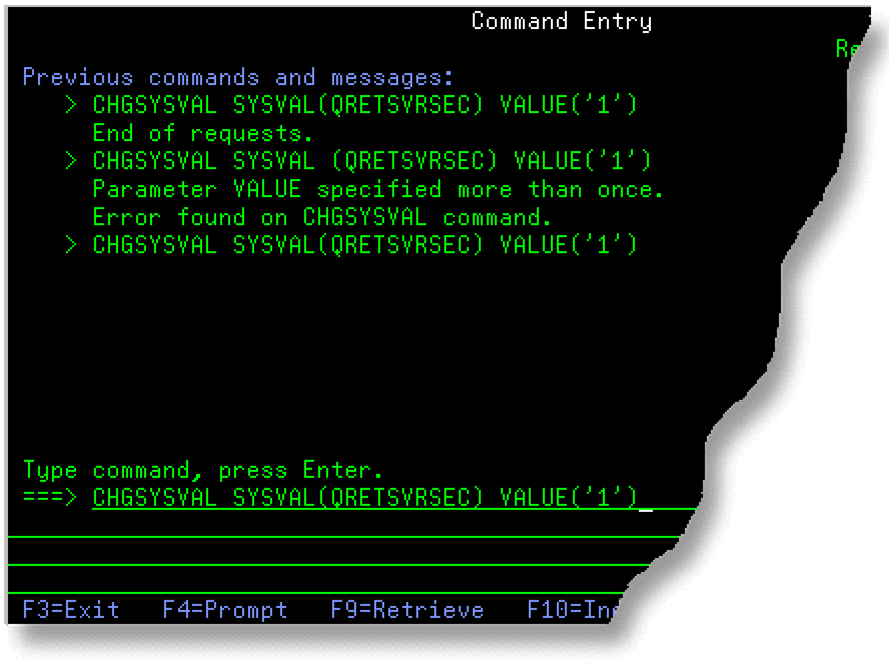
Set Retain Server Security Data Value to 1

The retain server security parameter determines whether your AS/400 will keep security data needed to authenticate users from different systems. When this value is set to 1 and a user signs on to the server, their security information is validated and kept on file. When your users sign on in the future, this data doesn't need to be authenticated, and gives them faster access to your private network.

To enable this feature:

1 On the AS/400 command line, type:

CHGSYSVAL SYSVAL (QRETSVRSEC) VALUE ('1').



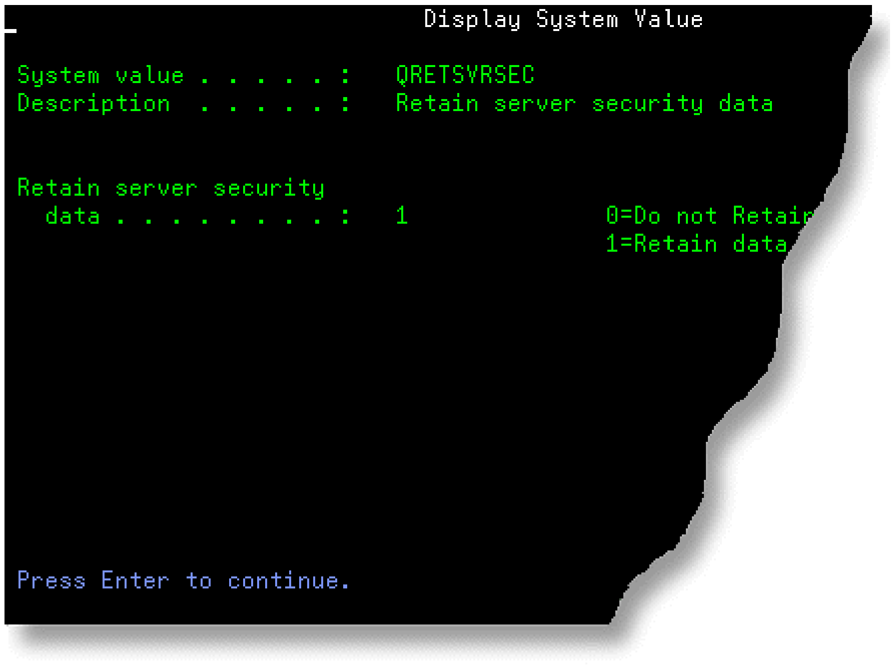
```
                                         Command Entry
                                                                          Re
 Previous commands and messages:
    > CHGSYSVAL SYSVAL(QRETSVRSEC) VALUE('1')
      End of requests.
    > CHGSYSVAL SYSVAL (QRETSVRSEC) VALUE('1')
      Parameter VALUE specified more than once.
      Error found on CHGSYSVAL command.
    > CHGSYSVAL SYSVAL(QRETSVRSEC) VALUE('1')







 Type command, press Enter.
 ===> CHGSYSVAL SYSVAL(QRETSVRSEC) VALUE('1')_



 F3=Exit   F4=Prompt   F9=Retrieve   F10=In
```

Press the **Enter key**.

2 To confirm the new value is **1**, on the AS/400 command line, type:

DSPSYSVAL SYSVAL (QRETSVRSEC).

The *Display System Values* window appears.



```
                              Display System Value

System value . . . . . :   QRETSVRSEC
Description  . . . . . :   Retain server security data


Retain server security
  data . . . . . . . . :   1              0=Do not Retai
                                          1=Retain data






Press Enter to continue.
```

You'll see a **1** in the **Retain server security data** field.

Now that your AS/400 has a retain server security data value of 1, it will store the security data needed to authenticate users from different systems.

Set Up AS/400 IPSec Policy

In this section, you'll set up IPSec policies to protect data that streams through your L2TP tunnels, with encryption and authentication; encryption makes stolen messages unreadable, and authentication ensures that incoming messages are sent by validated users.

To set up IPSec policies on your AS/400, you'll:

1. Build a Base IPSec Policy
2. Modify Key Policy
3. Modify Data Policy

Once your L2TP tunnels are protected through IPSec policies using encryption and authentication, you'll verify your AS/400 shuts down L2TP tunnels when VPN sessions end with the L2TP terminator.

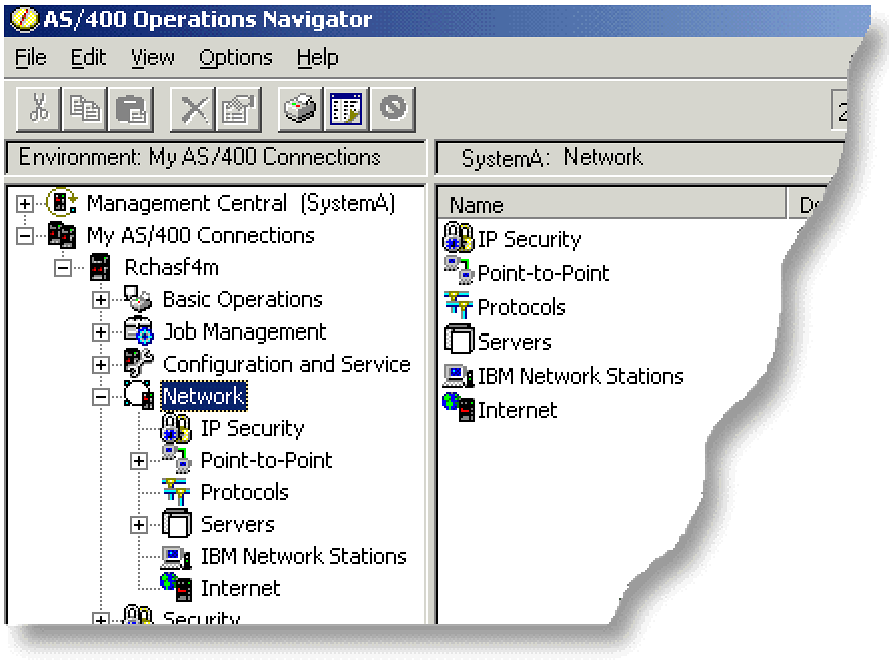
Build a Base IPSec Policy

IPSec policies define how you'll protect your keys and data during internet key exchange (IKE). During IKE, you'll transmit sensitive information that's used to establish a VPN session. It's critical you protect this information with IPSec.

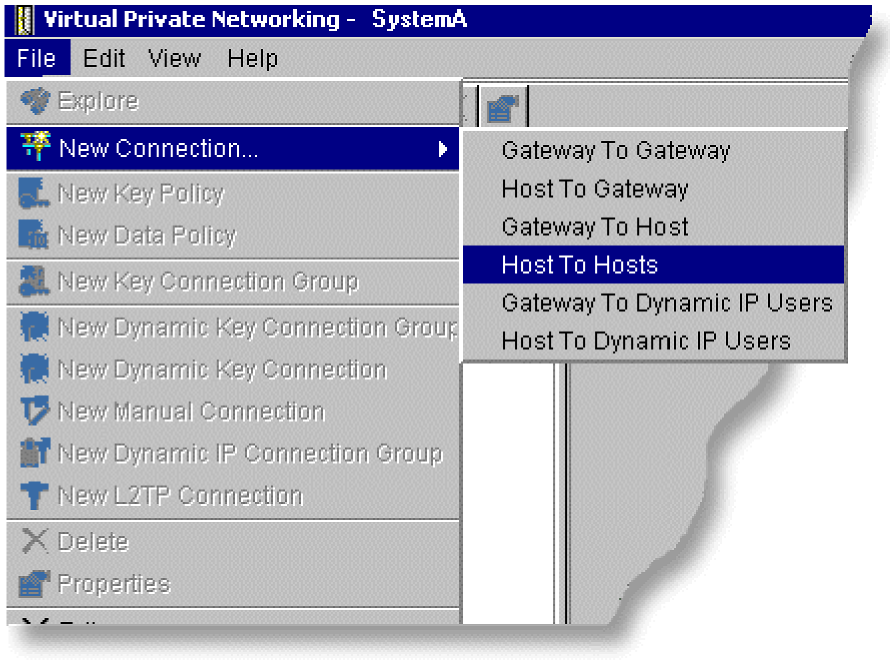
Note A secure key exchange is the most important factor in establishing a secure and private VPN connection. If your keys are compromised, then your authentication and encryption efforts, no matter how strong, become worthless.

To build a base IPSec policy, follow these steps:

1. In AS/400 Operations Navigator, double-click on **My AS/400 Connections**. The *My AS/400 Connections* window appears.
2. Double-click on your server (e.g., sysas400).
3. Double-click on **Network**. The *Network* window appears.



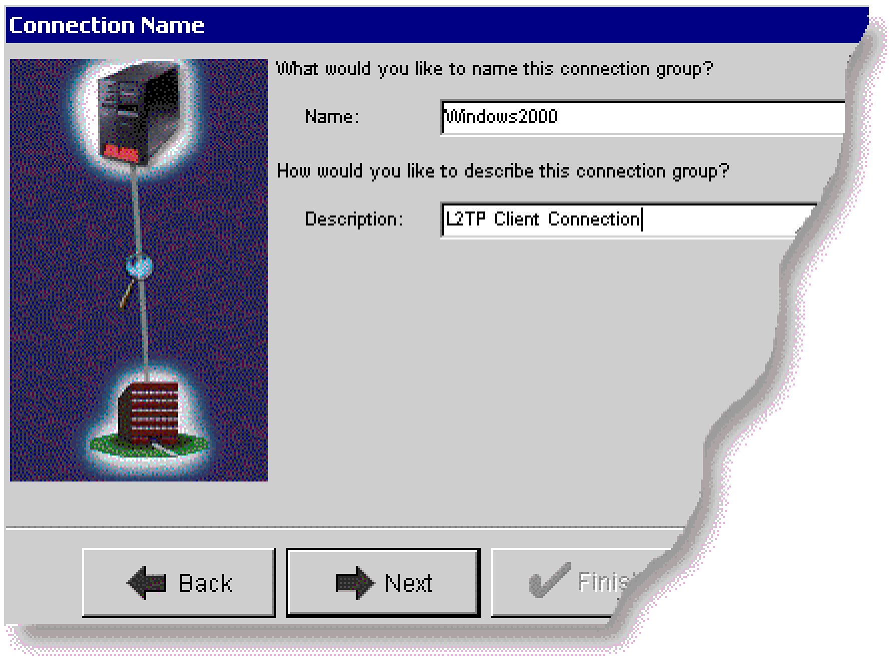
4. Double-click on **IP Security**. The *IP Security* window appears.
5. In the **IP Security** window, double-click on **Virtual Private Networking**. The *Virtual Private Networking* window appears.



6 In the **Virtual Private Networking** window, select **New Connection** from the *File* menu.

7 Select **Host To Hosts** from the pop-up menu. The *New Connection Wizar*d window appears.

8 Click on **Next**. The *Connection Name* window appears.


Connection Name

What would you like to name this connection group?

Name: Windows2000

How would you like to describe this connection group?

Description: L2TP Client Connection

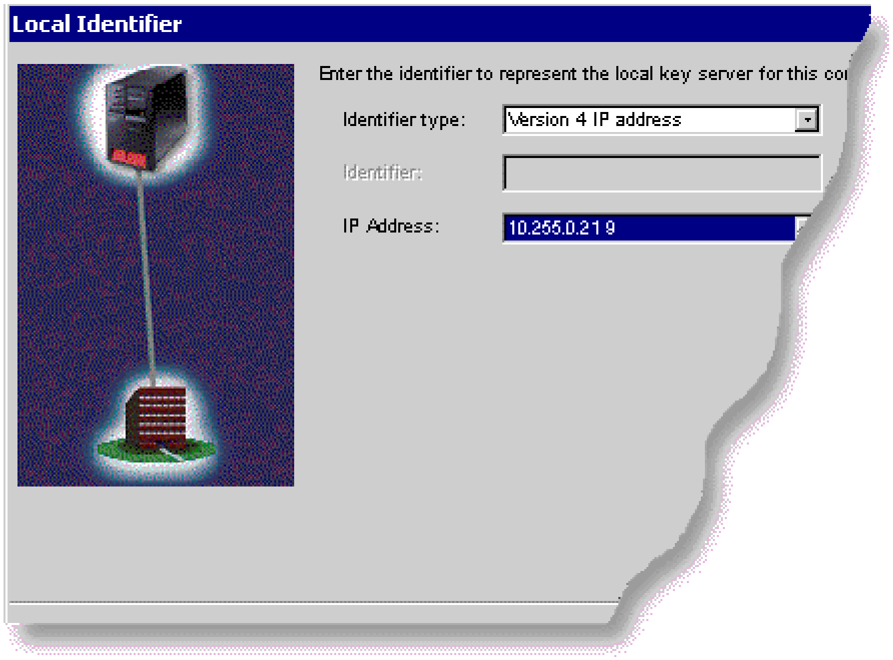
9 In the **Name** field, type a name for your connection group (e.g., Windows2000).

10 In the **Description** field, type **L2TP Client Connection**.

11 Click on **Next.** The *Key Policy* window appears.

12 Select **Highest security, lowest performance**.

13 Click on **Next**. The *Local Identifier* window appears.



Enter the identifier to represent the local key server for this co

Identifier type: Version 4 IP address

Identifier:

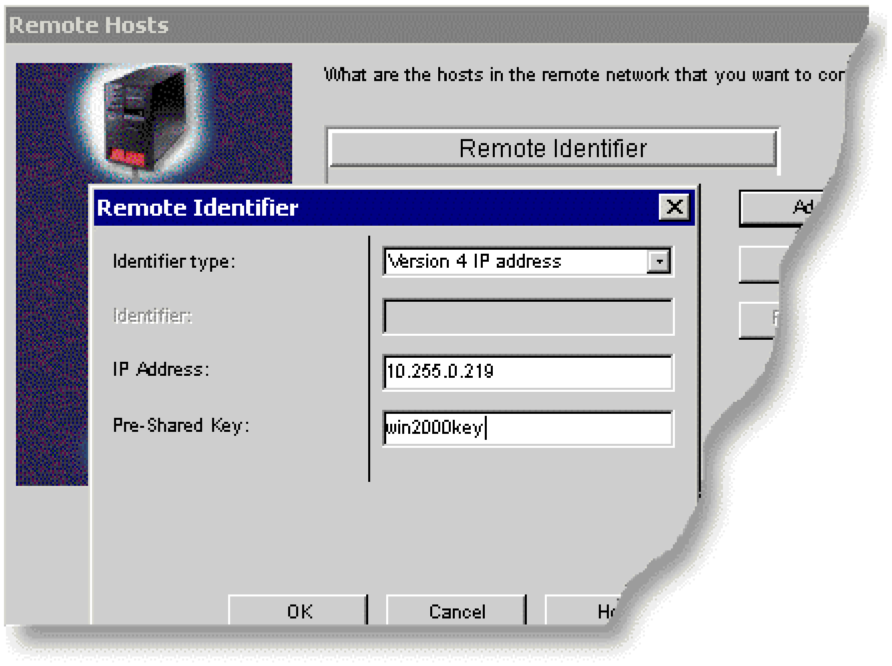
IP Address: 10.255.0.219

14 In the **Identifier type** field, select **Version 4 IP address** from the drop-down list.

15 In the **IP Address field**, select the IP address of the local key server making this connection (e.g., 10.255.0.219).

16 Click on **Next**. The *Remote Hosts* window appears.

17 Click on **Add**. The *Remote Identifier* window appears.


Remote Hosts

What are the hosts in the remote network that you want to cor

Remote Identifier

Identifier type: Version 4 IP address

Identifier:

IP Address: 10.255.0.219

Pre-Shared Key: win2000key

18 In the **Identifier type** field, select **Version 4 IP address** from the drop-down list.

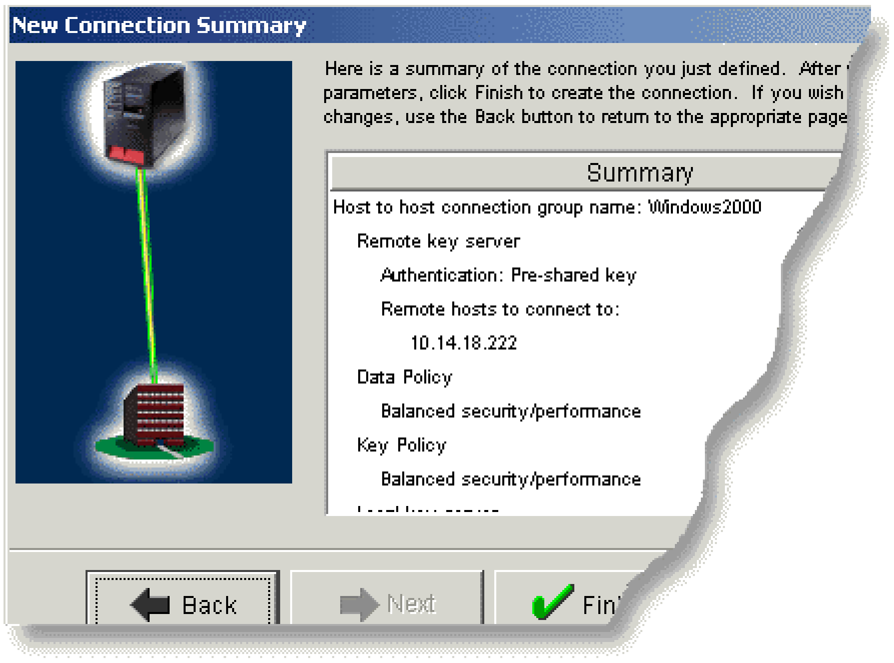
19 In the **IP Address** field, type the IP address of your AS/400 (e.g., 10.255.0.219).

20 In the **Pre-Shared Key** field, type **win2000key**.

21 Click on **OK**. The *Data Policy* menu appears.

22 Select **Balance security and performance**.

23 Click on **Next**. The *New Connection Summary* window appears. Make sure to verify your connection information.



Here is a summary of the connection you just defined. After
parameters, click Finish to create the connection. If you wish
changes, use the Back button to return to the appropriate page

Summary
Host to host connection group name: Windows2000
Remote key server
Authentication: Pre-shared key
Remote hosts to connect to:
10.14.18.222
Data Policy
Balanced security/performance
Key Policy
Balanced security/performance

24 Click on **Finish**.

Once you've built your IPSec policy, you'll modify the base IPSec policy so your VPN handles key exchanges.

Modify Key Policy

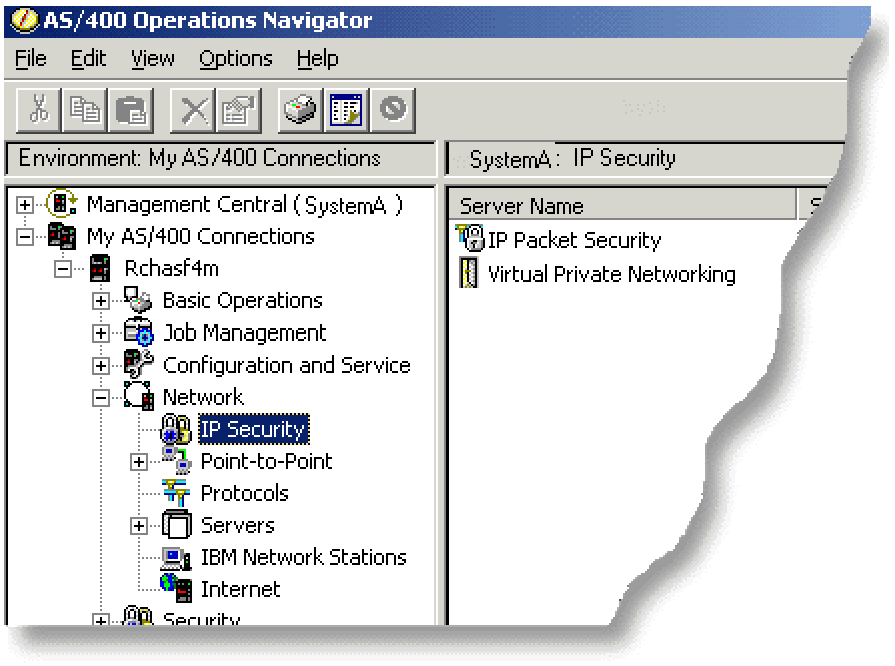
The key policy is necessary to protect initial communications between endpoints in a VPN sessions. You'll modify your key policy, so your AS/400 uses a pre-shared secret to generate cryptographic keys.

To modify your key policy, follow these steps:

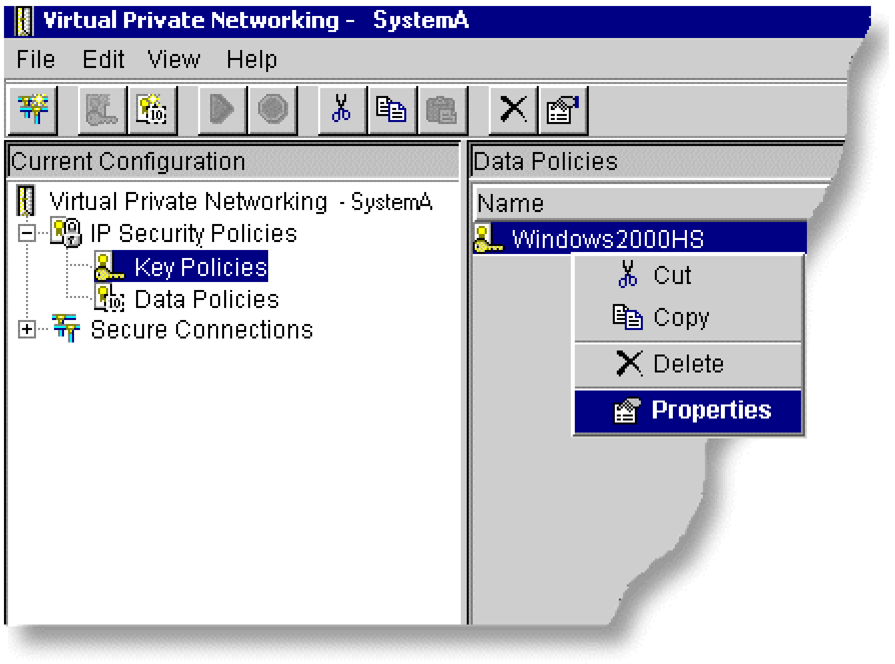
1 In AS/400 Operations Navigator, double-click on **My AS/400 Connections**. The *My AS/400 Connections* window appears.

2 Double-click on your server (e.g., sysas400).

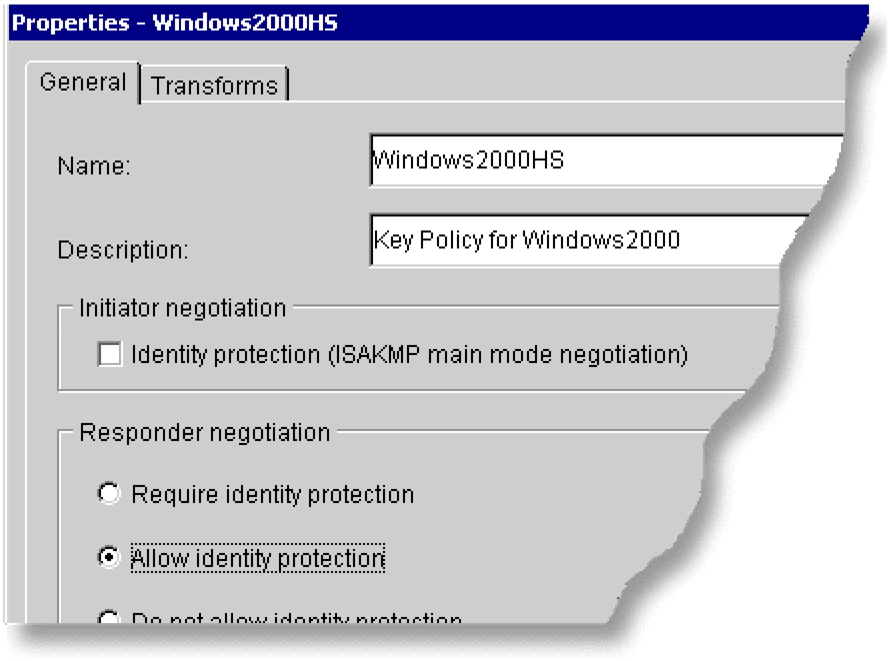
3 Click on **IP Security**. The *IP Security* window appears.



4 Double-click on **Virtual Private Networking**. The *Virtual Private Networking* window appears.



5 Right-click on **Windows2000HS**, and select **Properties** from the pop-up menu. The *Properties* window appears.



Properties - Windows2000HS

Name: Windows2000HS

Description: Key Policy for Windows2000

Initiator negotiation

[ ] Identity protection (ISAKMP main mode negotiation)

Responder negotiation

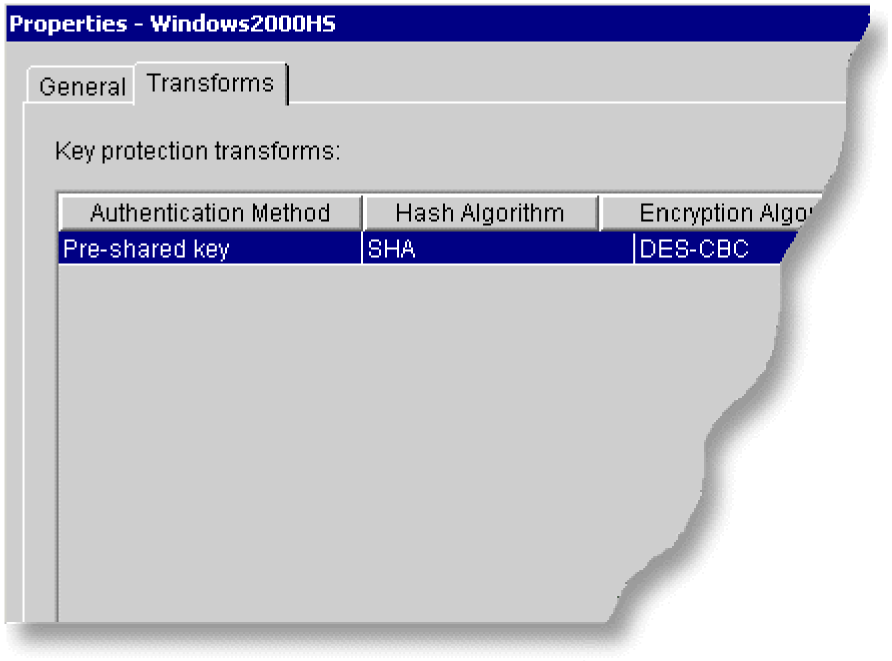
( ) Require identity protection

(●) Allow identity protection

( ) Do not allow identity protection

6 Select the **Allow identity protection** radio button from the **Responder negotiation** field.

7 Select the **Transforms** tab.



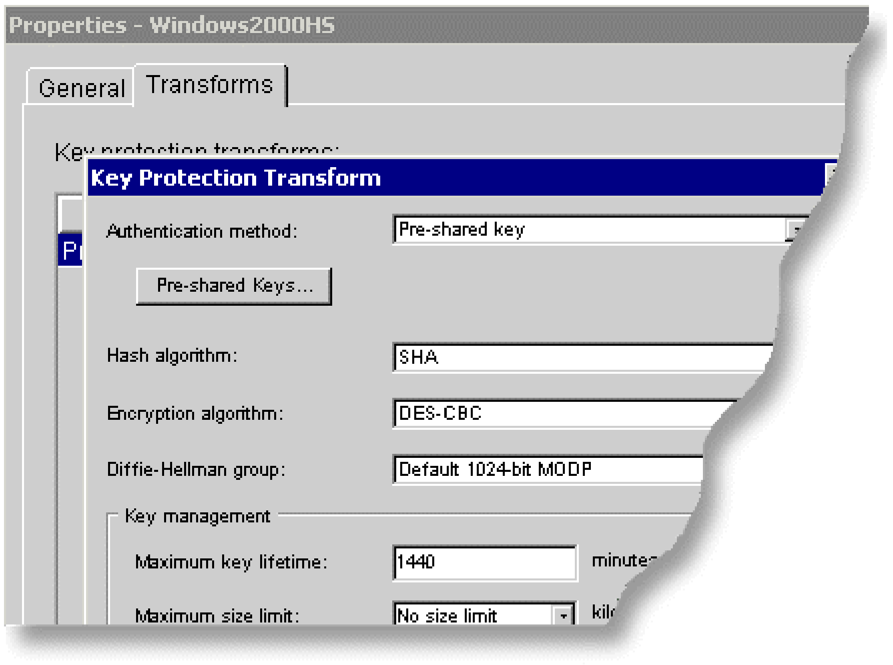
Properties - Windows2000HS

Key protection transforms:

Authentication Method	Hash Algorithm	Encryption Algo
Pre-shared key	SHA	DES-CBC

8 In the **Key protection transforms** field, highlight your key protection transforms entry (e.g., Pre-shared key).

9 Click on **Edit**. The *Key Protection Transform* window appears.



Authentication method: Pre-shared key
Hash algorithm: SHA
Encryption algorithm: DES-CBC
Diffie-Hellman group: Default 1024-bit MODP
Maximum key lifetime: 1440
Maximum size limit: No size limit

10 In the **Diffie-Hellman group** field, select **Default 1024-bit MODP** from the drop-down list.

11 Click on **OK** to close the *Key Protection Transforms* window.

12 Click on **OK** to save changes to your key policy.

Once you modify your key policy you'll modify your data policy, which protects data during the internet key exchange (IKE) in Phase II.

Modify Data Policy

The data policy tells your AS/400 how to protect data you send over the Internet. You'll modify your data policy, so your AS/400 generates cryptographic keys to protect data during Phase II negotiations.

To modify your data policy, follow these steps:

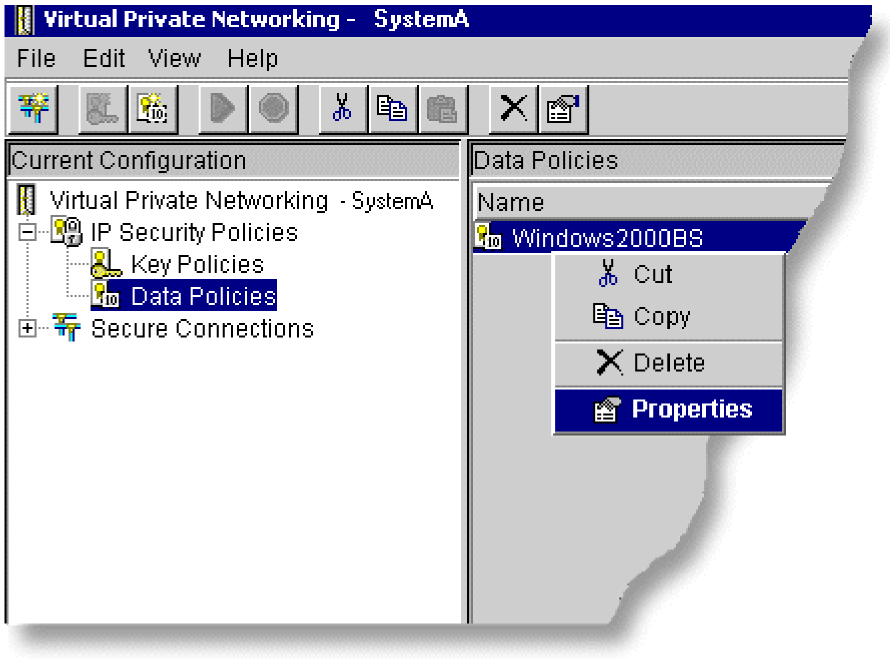
1 In AS/400 Operations Navigator, double-click on **My AS/400 Connections.** The *My AS/400 Connections* window appears.

2 Double-click on your server (e.g., sysas400).

3 Click on **IP Security**. The *IP Security* window appears.

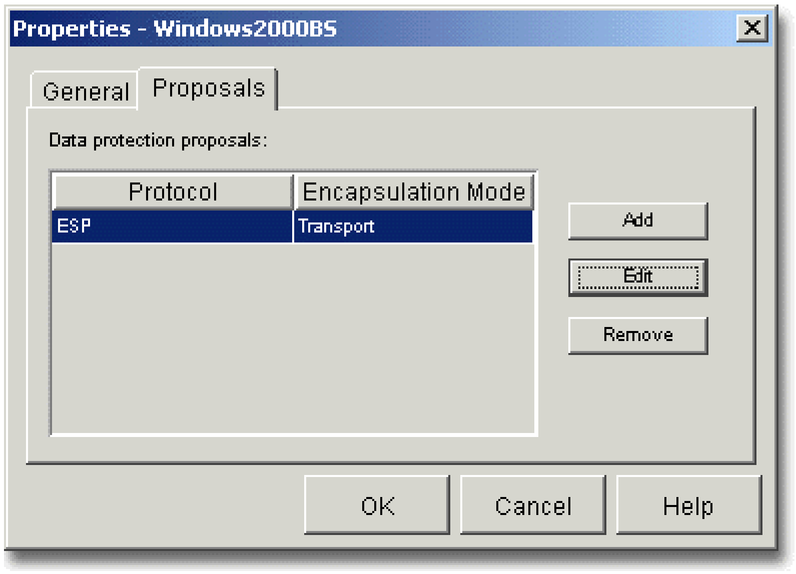
4 Double-click on **Virtual Private Networking**. The *Virtual Private Networking* window appears.

5 In the **Virtual Private Networking** window, click on **IP Security Policies**, and select **Data Policies**.



6 In the **Data Policies** window, right-click on **Windows2000BS** and select **Properties** from the pop-up menu. The *Properties* window appears.

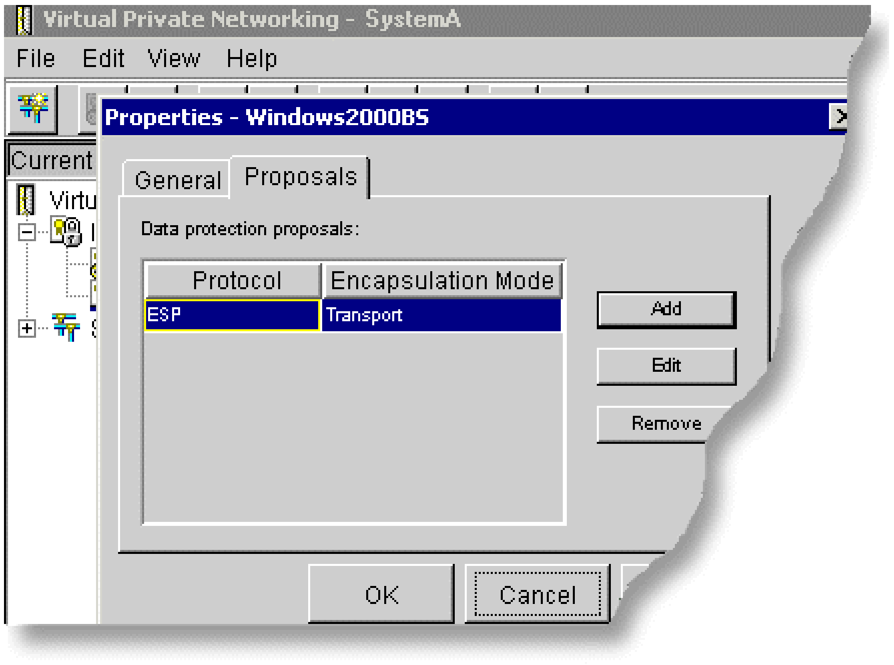
7 Select the **Proposals** tab.



Data protection proposals:

Protocol	Encapsulation Mode
ESP	Transport

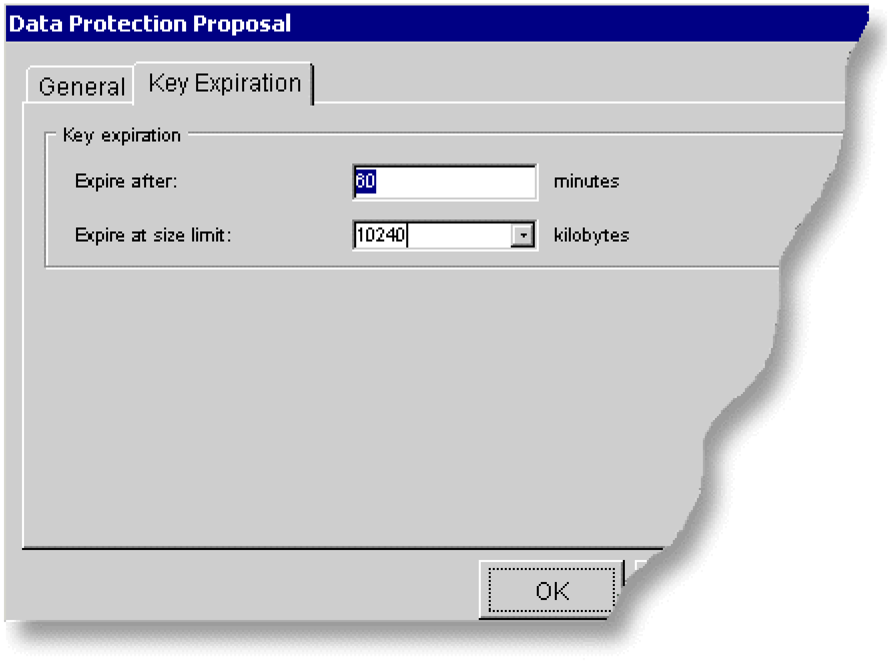
8 In the **Protocol** field, highlight **ESP**.



Data protection proposals:

Protocol	Encapsulation Mode
ESP	Transport

9 Click on **Edit**. The *Data Protection Proposal* window appears.



Expire after: 60 minutes

Expire at size limit: 10240 kilobytes

10 Select the **Key Expiration** tab.

11 In the **Expire at size limit** field, type **10240**.

12 Click on **OK** to close the *Data Protection Proposal* window.

13 Click on **OK** to save changes to the data policy.

Once you modify your data policy, you'll need to configure your AS/400 L2TP terminator.

Set Up Your IBM HTTP Server on AS/400 to Serve the Internet

This topic steps you through the process of setting up your AS/400 to serve Web pages to the Internet.

We've split the set up process into four steps:

1. Arrange Internet access with an ISP and record technical information <removed>
2. Adjust your TCP/IP settings
3. Start an administrative server instance <removed>
4. Serve Web pages from your AS/400 <removed>

✓ **Note** For this topic, we assume that you have a PC connected to your AS/400 with *IOSYSCFG and *ALLOBJ authority.

Adjust your TCP/IP settings

In this step, you'll equip AS/400 with TCP/IP so it will send and receive Web documents across the Internet. TCP/IP will also enable you to communicate across your local network as you work with the Administrative Server Instance in the next step.

1 Give Your AS/400 an IP Address

2 Define Local Domain and Host Names

3 Build a Local Host Table

4 Define a Default TCP/IP Route

5 Start TCP/IP Interface

Now that your IBM HTTP Server is equipped with TCP/IP, it can transmit information over the Internet. The next step is to put your IBM HTTP Server in an administrative mode (ADMIN Server Instance) that will enable you to make a few changes to the system.

Give Your AS/400 an IP Address

In this first step, you'll assign an IP Address to your AS/400 server, so other computers will recognize your server on the Internet:

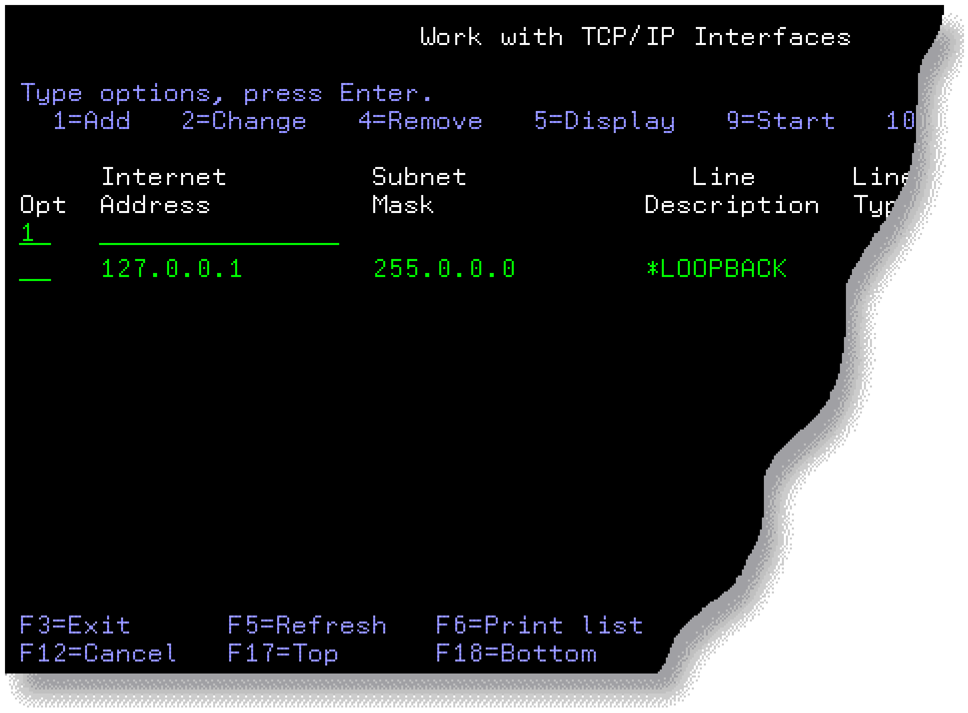
1 On the AS/400 command line, type:

CFGTCP

and press the **Enter** key. The *Configure TCP/IP* menu appears.

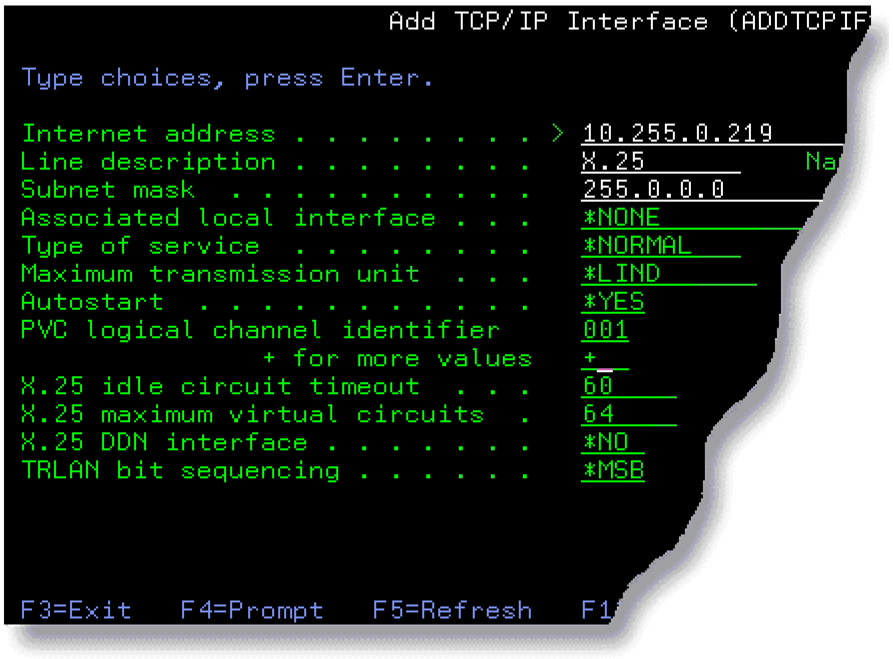
2 Enter **1** to Work with TCP/IP interfaces, then press the **Enter** key.

3 In the **Opt** field, type **1**, then press the **Enter** key:



```
                     Work with TCP/IP Interfaces

Type options, press Enter.
  1=Add    2=Change    4=Remove    5=Display    9=Start    10

     Internet            Subnet              Line        Line
Opt  Address             Mask                Description Typ
1    _______________
__   127.0.0.1           255.0.0.0           *LOOPBACK










F3=Exit       F5=Refresh    F6=Print list
F12=Cancel    F17=Top       F18=Bottom
```

The *Add TCP/IP Interface* screen appears:



```
                         Add TCP/IP Interface (ADDTCPIF

Type choices, press Enter.

Internet address . . . . . . . . . > 10.255.0.219
Line description . . . . . . . . .   X.25          Na
Subnet mask  . . . . . . . . . . .   255.0.0.0
Associated local interface . . .   *NONE
Type of service  . . . . . . . . .   *NORMAL
Maximum transmission unit  . . .   *LIND
Autostart  . . . . . . . . . . . .   *YES
PVC logical channel identifier     001
               + for more values   +
X.25 idle circuit timeout  . . .   60
X.25 maximum virtual circuits  .   64
X.25 DDN interface . . . . . . .   *NO
TRLAN bit sequencing . . . . . .   *MSB



F3=Exit    F4=Prompt    F5=Refresh    F1
```

4 In the **Internet address**, **Line description**, and **Subnet mask** fields, enter the information you collected from your ISP. For example:

- Internet address 10.255.0.219
- Line description X.25
- Subnet mask 255.0.0.0

Press the Enter key to continue.

Now that you've assigned an IP Address, you'll give your AS/400 Server a host name and a domain name.

Define Local Domain and Host Names

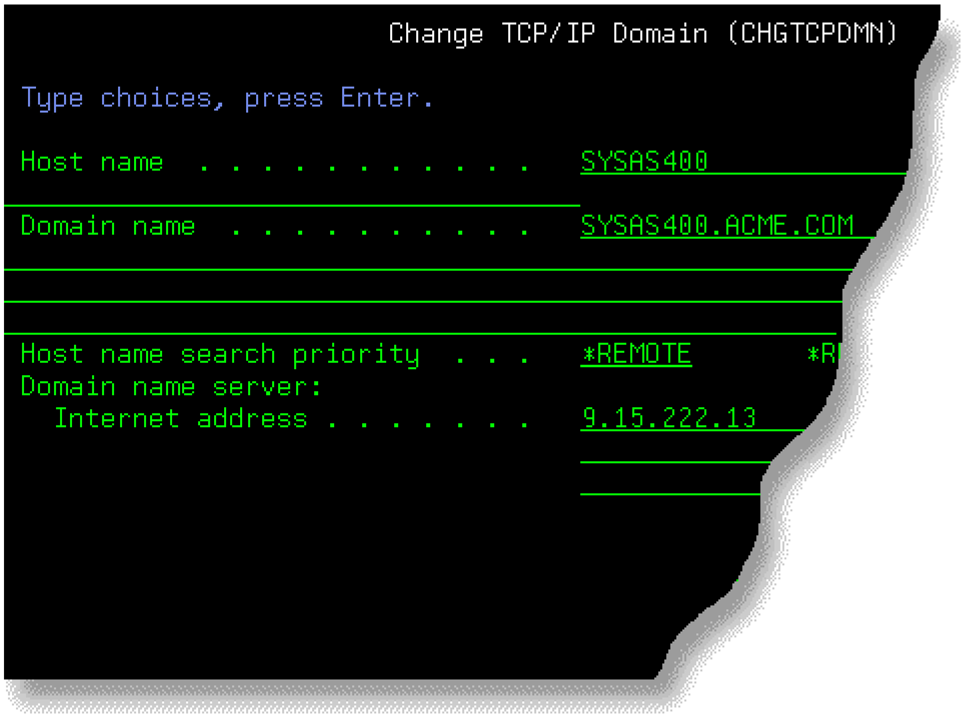
In this step, you'll give your AS/400 domain and host names, so your system is recognizable within your organization and on the Internet.

1 On the AS/400 command line, type:

CFGTCP

and press the **Enter** key. The *Configure TCP/IP* menu appears.

2 Select menu option **12** to change your TCP/IP Domain. Press the **Enter** key. The *Change TCP/IP Domain* screen appears:



```
                        Change TCP/IP Domain (CHGTCPDMN)

 Type choices, press Enter.

 Host name  . . . . . . . . . . .   SYSAS400

 Domain name  . . . . . . . . . .   SYSAS400.ACME.COM



 Host name search priority  . . .   *REMOTE       *R
 Domain name server:
   Internet address . . . . . . .   9.15.222.13
```

3 In the **Host name** field, enter the local host name for your AS/400 (e.g. SYSAS400) which you recorded in the Setup Parameters Form earlier.

4 In the **Domain name** field, enter the local domain name (e.g. SYSAS400.ACME.COM).

5 Press the **Enter** key and you're finished with this step.

Your next step is to build a local host table which enables AS/400 to associate your IP address with your domain and host names.

Build a Local Host Table

The local host table stores your AS/400's Internet (IP) address and its associated host name. When a user connects to the server using your host name, AS/400 finds the IP address associated with the name in the local host table. This is how users connect to your AS/400 over the Internet without knowing your IP address.

To build a local host table on your AS/400, follow these steps:

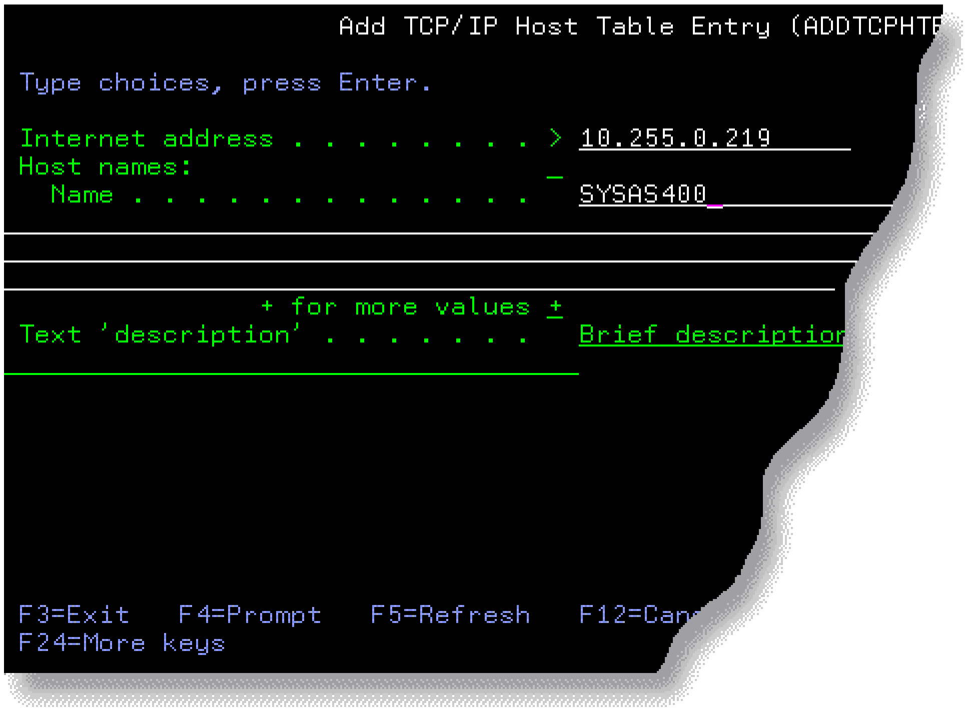
1 On the AS/400 command line, type:

ADDTCPHTE

and press the **Enter** key. The *Add TCP/IP Host Table Entry* screen appears.

2 In the **Internet address** field, enter the IP address provided by your ISP (e.g. 10.255.0.219). In the host names: **Name field**, enter your server's host name (e.g. SYSAS400).

Also, enter a plus sign (+) to the right of **+ for more values** as illustrated below. The plus sign indicates that you will define additional host names for your AS/400.



```
                     Add TCP/IP Host Table Entry (ADDTCPHTE

Type choices, press Enter.

Internet address . . . . . . . . > 10.255.0.219
Host names:                        _
  Name . . . . . . . . . . . . .   SYSAS400



                + for more values +
Text 'description' . . . . . . .   Brief description






F3=Exit   F4=Prompt   F5=Refresh   F12=Can
F24=More keys
```

3 Press the Enter key:

4 On the display that appears, enter your fully qualified domain name in the **Host names**: Name field. Your AS/400's fully qualified domain name is the host name followed by your local domain name (e.g. **SYSAS400.ACME.COM**):

Press the **Enter key** to continue.

Your next step is to assign a primary IP address to your AS/400 where all transmissions are sent as they journey onto the Internet.

Define a Default TCP/IP Route

Your AS/400 must connect to an ISP to reach the Internet. Here, you'll define the default route AS/400 will use to reach your ISP, which is your ISP's IP address.

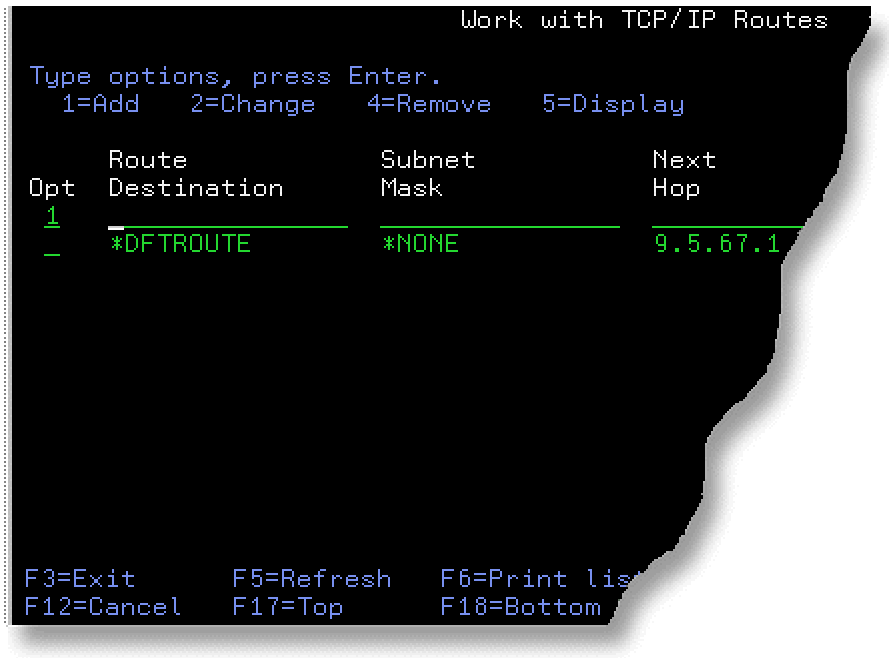
To define the default TCP/IP route, follow these steps:

1 On the AS/400 command line, type:

CFGTCP

and press the **Enter** key.

2 Select menu option **2** to work with TCP/IP routes. Press the **Enter** key. The *Work with TCP/IP Routes* screen appears:



```
                                   Work with TCP/IP Routes

 Type options, press Enter.
   1=Add    2=Change    4=Remove    5=Display

      Route                Subnet               Next
 Opt  Destination          Mask                 Hop
  1   _______________      _______________      _______
  _   *DFTROUTE            *NONE                9.5.67.1






 F3=Exit       F5=Refresh    F6=Print lis
 F12=Cancel    F17=Top       F18=Bottom
```

3 Enter a **1** in the **Opt** field to add a TCP/IP Route, and press the **Enter** key. The *Add TCP/IP Route* screen appears:

Refer to your completed Setup Parameters Form for this information:

4 In the **Route destination** field, type ***DFTROUTE**.

5 In the **Subnet mask** field, type your subnet mask (e.g. 255.255.255.0).

6 In the **Type of service** field, type ***NORMAL**.

7 In the **Next hop** field, type your *ISP's IP address* (e.g., 9.5.67.1).

8 In the **Maximum transmission unit** field, type **576** unless your ISP gives you a different value.

9 Press the **Enter** key.

10 Press **F3** to return to the *Configure TCP/IP* menu.

Next, you'll start your TCP/IP interface.

Start TCP/IP Interface

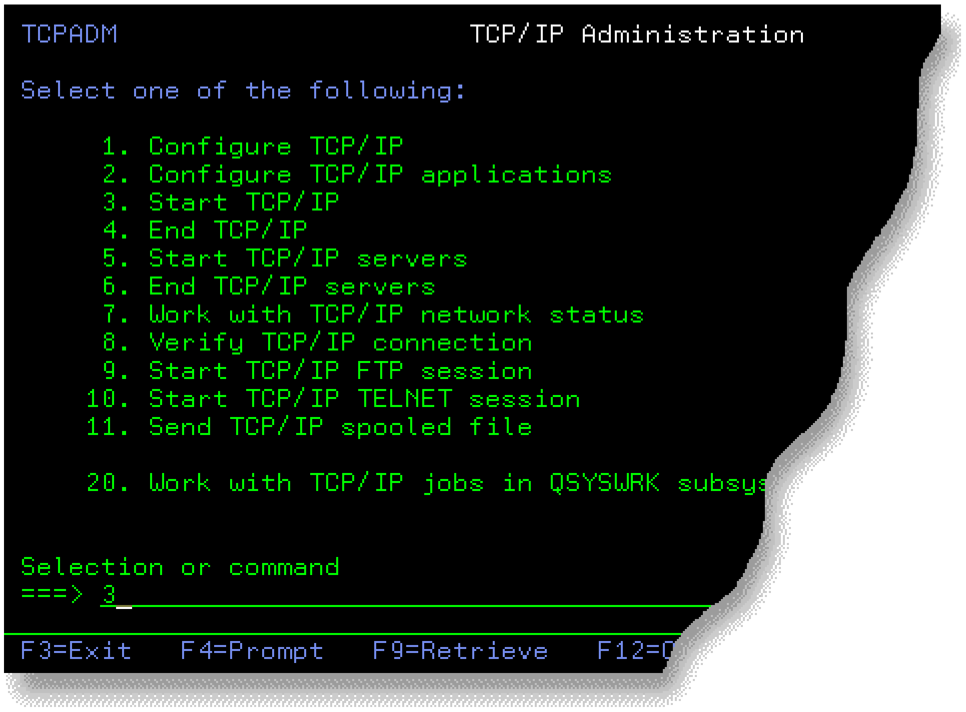
The TCP/IP Interface must be running before you can set up your administration server in the next step.

Just follow these steps to start your TCP/IP Interface:

1 On the AS/400 command line, type:

GO TCPADM

Press the **Enter** key. The *TCP/IP Administration* page loads:



```
TCPADM                             TCP/IP Administration

Select one of the following:

     1. Configure TCP/IP
     2. Configure TCP/IP applications
     3. Start TCP/IP
     4. End TCP/IP
     5. Start TCP/IP servers
     6. End TCP/IP servers
     7. Work with TCP/IP network status
     8. Verify TCP/IP connection
     9. Start TCP/IP FTP session
    10. Start TCP/IP TELNET session
    11. Send TCP/IP spooled file

    20. Work with TCP/IP jobs in QSYSWRK subsys


Selection or command
===> 3_

F3=Exit   F4=Prompt   F9=Retrieve   F12=C
```

2 Select option **3** (Start TCP/IP) from the *TCP/IP Administration* menu, and press the **Enter** key.

3 When the display asks if you want to "Start application servers" and "Start TCP/IP interfaces" make sure each parameter states ***YES**.

Press the **Enter** key. The message "TCP/IP started" appears.

✓ **Note** If you power off and then power on (IPL) your AS/400, you need to repeat this step in order to start TCP/IP again, because AS/400 does not auto-start the TCP/IP server at time of IPL.

Now that TCP/IP is configured and your application server is running, it's time to set up the administration server, so you can make further changes to system settings.